INCOMPRESSIBLE FLUID SIMULATION AND ADVANCED SURFACE HANDLING WITH SPH

A dissertation submitted to
Faculty of Economics, Business Administration and Information Technology of
the University of Zurich

for the degree of
Doctor of Sciences

presented by
Barbara Solenthaler
Master of Science in Computer Science, ETH Zurich

accepted on the recommendation of
Prof. Dr. Renato Pajarola, University of Zurich, Switzerland, examiner
Prof. Dr. Markus Gross, ETH Zurich, Switzerland, co-examiner

January 2010

The Faculty of Economics, Business Administration and Information Technology of the University of Zurich herewith permits the publication of the aforementioned dissertation without expressing any opinion on the views contained therein.

Zurich, October 21, 2009

Head of the Ph.D. program: Prof. Abraham Bernstein, Ph.D.

ABSTRACT

Particle-based fluid simulations have become popular in computer graphics due to their natural ability to handle free surfaces and interfaces, splashes and droplets, as well as interaction with complex boundaries. However, particle methods have some disadvantageous properties degrading the physical behavior of a simulated fluid and thus the resulting visual quality. Although these problems are present in almost any particle-based fluid solver, this dissertation addresses some of the major problems of the Lagrangian method Smoothed Particle Hydrodynamics (SPH).

This thesis starts by reviewing the standard SPH model and its difficulties to satisfy the incompressibility condition. In the standard model, liquids are typically approximated by compressible fluids where pressures are determined by an equation of state, resulting in undesired compression artifacts. Although incompressibility can be enforced, it represents the most expensive part of the whole simulation process and thus renders particle methods less attractive for high quality and photorealistic water animations. In this thesis, we present a novel, incompressible fluid simulation method based on SPH. In our method, incompressibility is enforced by using a prediction-correction scheme to determine the particle pressures. For this, the information about density fluctuations is actively propagated through the fluid and pressure values are updated until the targeted density is satisfied. With this approach, the costs per simulation update step can be held low while still being able to use large time steps in the simulation.

Next, we shift our attention to the problem of complex interactions between multiple different fluids as well as between fluids and solids. We first focus on

the artifacts caused by standard SPH when simulating multiple fluids with high density ratios. In the standard model, the smoothed quantities of particles near the fluid interface show falsified values and the physical behavior is severely affected, especially if density ratios become large. The artifacts include spurious and unphysical interface tension as well as severe numerical instabilities. In this thesis we derive a formulation which can handle discontinuities at interfaces of multiple fluids correctly and thus avoids the problems present in standard SPH. With our concepts, an animator has full control over the behavior of multiple interacting fluids.

Furthermore, we propose to represent both, fluids and solids, by particles, facilitating the interaction between the different object types. We present a unified simulation model for fluids, rigid, and elastic objects, and show how phase transitions can be modeled by only changing the attribute values of the underlying particles. New effects like merging and splitting due to melting and solidification are demonstrated, and we show that our model is able to handle coarsely sampled and even coplanar particle configurations without further treatment.

Finally, we present a novel point refinement method to achieve a higher visual quality of low-resolution fluids. We introduce new algorithms to efficiently upsample an initial point set given by the physical computation. Our method features the ability to accurately preserve surface details and to reach a uniform point distribution. Another challenge is to reconstruct smooth surfaces from the particles. The visualized fluids typically suffer from bumpy surfaces related to the irregular particle distribution. In order to achieve smooth surfaces, this thesis introduces a new surface reconstruction technique based on the center of mass of the particle neighborhood. We show how artifacts in concave regions can be avoided by considering the movement of the center of mass.

ACKNOWLEDGEMENTS

During my PhD studies I was surrounded by many great people who supported me in my work.

First of all, I would like to thank my advisor Renato Pajarola for giving me the opportunity to pursue my PhD studies at the University of Zurich. I'm very grateful for his support, and I was able to profit greatly from his knowledge and experience. I would also like to thank Markus Gross for accepting to be part of my committee. Already during my master studies at ETH Zurich, Markus showed me the joy of research in computer graphics and animation, which greatly influenced my future research projects during my doctoral program.

Many thanks go to my colleagues from the VMML lab at University of Zurich and the IVT lab at ETH Zurich. They supported me with many helpful discussions about research and simulation, encouraging comments during deadline periods, and great sports and leisure activities.

CONTENTS

NOTATIONS

Acronyms

CFL	Courant-Friedrichs-Levy
CSPH	Corrected SPH
EOS	Equation of state
EV	Eigenvalue
FDM	Finite Difference Method
FEM	Finite Element Method
FLIP	Fluid-Implicit-Particle Method
FPS	Frames per second
GPU	Graphics Processing Unit
ISPH	Incompressible SPH
LBM	Lattice Boltzmann Method
LJ	Lennard-Jones
MLS	Moving Least-Squares
MPS	Moving Particle Semi-Implicit Method
MSS	Mass spring systems
NS	Navier-Stokes
PCISPH	Predictive Corrective Incompressible SPH
PIC	Particle In Cell Method
SPH	Smoothed Particle Hydrodynamics
WCSPH	Weakly Compressible SPH

Operators and Functions

$\mathbf{x}_{ij}$	distance vector between particle i and j
$\nabla\cdot$	divergence operator: $\nabla\cdot = \frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}$
∇	gradient operator: $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$
∇^2	Laplacian operator: $\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$
$\phi(\mathbf{x})$	implicit surface function
$W(\mathbf{x}_{ij}, h)$	kernel function

Physics

Δt	simulation time step
η	density fluctuation threshold
$\mathbf{F}$	force
$\mathbf{f}$	force density
$\mathbf{g}$	gravity
μ	viscosity constant
ρ_0	rest density
θ	tension coefficient
c_s	speed of sound
E	Young's modulus
h	support radius or smoothing length
k	gas constant or stiffness value
t	time
T_{melt}	melting temperature
T_{solid}	solidification temperature
v	Poisson's ratio

Fluid Particles

$< c >_i$	smoothed color value
δ_i	particle density
κ_i	curvature
$\mathbf{a}_i$	acceleration
$\mathbf{F}_i^{external}$	external forces acting on i
$\mathbf{F}_i^{interface}$	interface tension force acting on i
$\mathbf{F}_i^{pressure}$	pressure force acting on i
$\mathbf{F}_i^{surface}$	surface tension force acting on i

$\mathbf{F}_i^{viscosity}$	viscous force acting on i
$\mathbf{n}_i$	normal
$\mathbf{v}_i$	velocity
$\mathbf{v}_i^*$	predicted velocity
$\mathbf{x}_i$	position
$\mathbf{x}_i^*$	predicted position
$\rho_{err_i}^*$	predicted density variation
ρ_i	density
ρ_i^*	predicted density
$\tilde{\rho}_i$	modified density
$\tilde{p}_i$	modified pressure
c_i	color value
i	particle i
m_i	mass
N_i	neighborhood of i
p_i	pressure
T_i	temperature
V_i	volume

Solid Particles

ϵ_i	strain tensor
$\mathbf{F}_i^{elastic}$	elastic forces acting on i
$\mathbf{I}$	inertia tensor
$\mathbf{L}$	angular momentum
ω	angular velocity
$\overline{V}$	body volume or reference volume
σ_i	stress tensor
τ_i	torque
τ_{body}	torque of a rigid body
U_i	strain energy

Surface Points

$\mathbf{N}_i^{surface}$	surface point neighborhood
$\mathbf{n}_p$	point normal
$\mathbf{x}_p$	point position
P_i	refined point set
r_i	refinement step radius

s_i	refinement step
w_p	point weight

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Physics-Based Fluid Simulation

Animating fluids by physics-based simulation has gained increasing importance in computer graphics in the last decade. Application areas of fluid simulations include feature films, commercial work, medical simulations, virtual environments, and computer games (Figure 1.1). The characteristics of the simulation methods employed in the individual fields of application are vastly different. In the film industry, the simulated fluids visually have to match reality so that an observer cannot distinguish between real scenarios and completely modeled and simulated environments. The inclusion of simulated fluids is a fairly new development that has been advanced by the increase in processing power over the last years. Hence, the film industry is making more and more use of simulation tools to model certain fluid effects that for an observer appear to be physically plausible. To achieve the desired realism high resolution fluid simulations have to be run that capture small-scale features like splashes, droplets and foam [Losasso et al., 2008]. This represents a very time-consuming process, thus the fluid movement is typically computed offline by using several CPUs in parallel. In contrast to the film and television industry, virtual reality applications and computer games are focusing on speed and stability of the simulation. The complexity of the physics is often reduced such that a fluid solver can be run in real-time on standard PCs and game consoles. This comes at the cost of low-resolution fluids and simplified physics, degrading the resulting physical behavior of the fluid. In particle-based fluid sim-

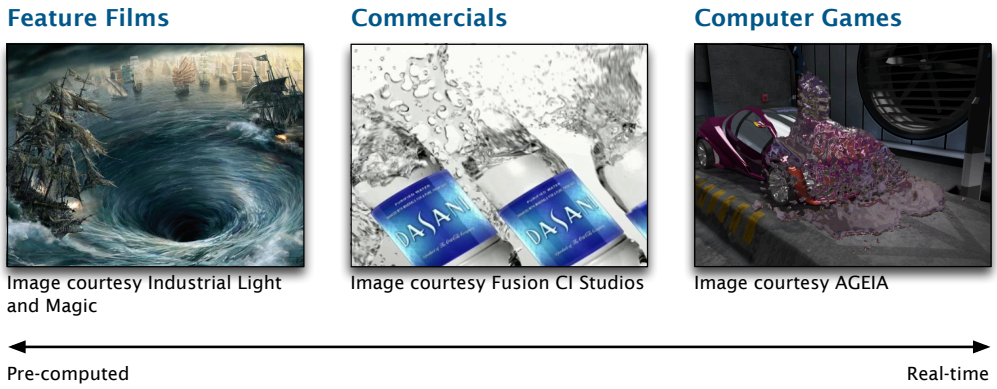


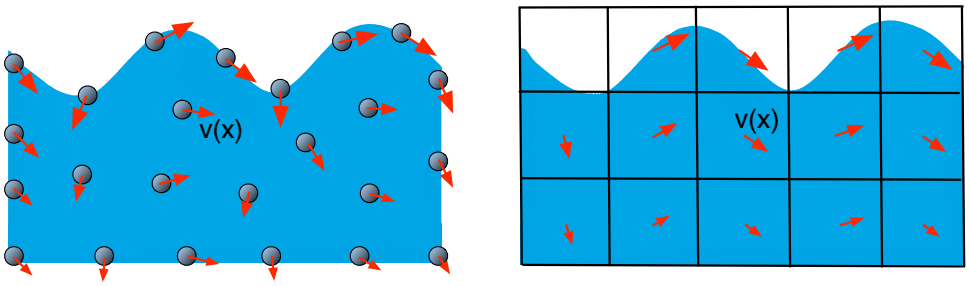
Figure 1.1: *Applications of fluid simulations in computer graphics include the film industry (left), commercials (middle), and computer games (right). The characteristics of a simulation highly depends on the particular application.*

ulations, such simplifications include for example that liquids are approximated by compressible matter in order to be able to increase the time step of the simulation and thus the performance [Müller et al., 2003]. The performance can be further increased by reducing the number of particles which discretize the fluid's volume. The real-time constraint also requires improved methods for rendering the fluids, for example by taking the visibility and view-dependent level of details into account [Müller et al., 2007].

Although the different fields of applications demand different simulation properties, they have the common need for new and efficient methods that improve the fluid behavior and thus increase the resulting visual quality. This thesis concentrates on such improvements for the particular Smoothed Particle Hydrodynamics (SPH) fluid solver.

1.2 Lagrangian and Eulerian Models

In the graphics literature, there are currently two main approaches to simulating fluids, the Eulerian and the Lagrangian models. The underlying representations of both techniques are illustrated in Figure 1.2. Figure 1.3 gives an overview over the main characteristics of both approaches. In the Eulerian grid approach (recent papers include [Stam, 1999; Foster and Fedkiw, 2001; Enright et al., 2002; Feldman et al., 2005; Losasso et al., 2006a; Losasso et al., 2008]), it is observed how fluid quantities change in time at fixed points in space. In the MAC grid method, the first step is to identify which grid cells currently contain fluid, and a buffer zone of air is created around the fluid. After the grid has been updated, the velocity field is



(a) Lagrangian Viewpoint (b) Eulerian Viewpoint

Figure 1.2: *In the Lagrangian viewpoint (a), the fluid is discretized by particles carrying physical quantities like velocity* $\mathbf{v}(\mathbf{x})$*. In the Eulerian representation (b), the quantities are measured at fixed points in space, for example at the cell centers. Note that staggering the velocity components by placing them to the center of the cell faces tends to produce more stable results.*

advanced by Δt. Then, external forces and viscosity are added. At this point, the velocity field does not satisfy the incompressibility condition, thus the pressures in the fluid cells are set so that the divergence throughout the fluid will be zero. To make all cells divergence free after pressure has been applied, a large, sparse linear system with one variable for the pressure of each cell containing fluid has to be solved. After the pressure has been applied, the fluid velocities are extrapolated into the buffer zone surrounding the fluid, and the velocities of solid cells are set. After the velocity field has been advanced, the marker particles or some other representation of the fluid such as a level set can be moved.

One strength of grid-based methods is the solution of the incompressibility condition. The grid discretization facilitates the formulation and solution of the equation system to determine the pressure values, thus incompressibility can be enforced within reasonable computation time. However, the grid discretization makes it difficult to accurately handle complex boundaries, free surfaces and interfaces between multiple fluids, as well as splashes and single droplets [Chentanez et al., 2006; Losasso et al., 2006b]. Other difficulties exist with the advection part of the Navier-Stokes (NS) equations often resulting in numerical dissipation due to averaging operations [Molemaker et al., 2008]. Visually, this leads to undesired damping, and often nonphysical terms such as vorticity confinement must be added to counter this effect [Selle et al., 2005]. Recently, [Mullen et al., 2009] proposed a family of fully Eulerian integration schemes that provide control over the amount of dissipation.

On the other hand, Lagrangian particle approaches trivially handle advection

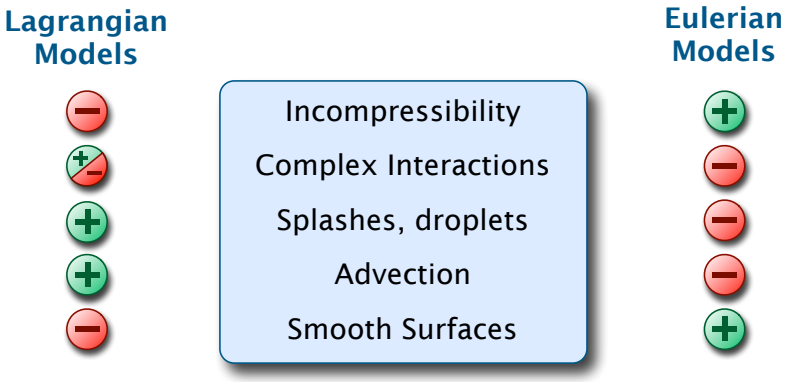


Figure 1.3: *Characteristics of Lagrangian and Eulerian fluid simulation models. In this thesis we use a Lagrangian fluid solver based on SPH, focusing on improving some major problems of particle-based methods.*

since the fluid is discretized by particles moving freely along the velocity field. In these meshfree methods, each particle is either directly associated with one discrete physical object or represents a part of the continuum problem domain. Hence the particle size can range from nano- to macro-scales. While molecular dynamics relate each particle to one molecule, a particle in SPH represents a certain amount of volume and thus SPH is employed to simulate macro areas where classical continuum assumptions apply.

The advantage of particles is the natural ability to handle free surfaces and interfaces, splashes and droplets, as well as the interaction with highly complex boundaries and solid objects [Müller et al., 2003; Müller et al., 2005b; Keiser et al., 2005; Solenthaler et al., 2007a; Solenthaler and Pajarola, 2008; Becker et al., 2009]. However, particle approaches have difficulties to satisfy the incompressibility condition of liquids such as water, since the fluid is typically approximated by a compressible fluid where pressures are determined by an equation of state (EOS) [Monaghan, 1994; Becker and Teschner, 2007]. This problematic issue, and the difficulties to reconstruct smooth surfaces out of particles, are the main reasons why grid methods are currently favored in graphics. However, particles and point representations have gained increasing attention in recent years, both for physics-based animations and rendering. Their simplicity and flexibility render a fluid solver into a powerful tool to simulate all kinds of phenomena such as water, smoke and fire, as well as complex interaction effects between multiple fluids and solid objects. Due to these beneficial properties, this thesis uses a Lagrangian fluid solver based on the SPH particle method.

1.3 Challenges of Particle Systems

As indicated above, particle systems feature some disadvantageous properties degrading the physical behavior and thus the resulting visual quality of a liquid. Although the described problems are present in almost any particle system, we specifically discuss some of the major problems of the SPH particle method which is the fluid solver this thesis is based on. In the list below, the bold items indicate problems that are addressed in this thesis. A short discussion about our proposed solutions of these problems will be given in Section 1.4.

1) *Comparison with Eulerian solvers*
Lagrangian SPH solvers and Eulerian grid solvers are two simulation models that are often used to simulate 3D fluids. In computer graphics, neither computational nor visual comparisons of these two solvers have been made so far. Hence it is not clear how the number of cells in grid solvers and the number of particles in SPH simulations actually relate to each other. Thus, the visual results and performance measurements of both methods can hardly be compared with each other. In order to evaluate the adequate solver for a particular problem it would be desirable to have better knowledge about the strengths and weaknesses of both solvers, including comparisons about fluid resolution, performance, and reproduced details of real fluids.

2) *Efficiency*
Fluid simulations require a high discretization resolution in order to achieve a certain physical and visual quality. Generally, the more particles that are used the lower the fluid viscosity and the more small-scale details like splashes, spray, and surface waves are visible. Most recent SPH research papers refer to particle numbers of up to 1-3M that are computed on a single CPU [Solenthaler and Pajarola, 2008; Becker et al., 2009; Solenthaler and Pajarola, 2009]. In order to speed up the computation of high resolution simulations, SPH has been implemented on the GPU with promising results [Harada et al., 2007; Zhang et al., 2008]. Another possibility is to reduce the overall particle number by adaptively sampling the fluid according to certain criteria like geometric complexity [Adams et al., 2007]. Although such an adaptive sampling reduces the computational costs considerably, difficulties exist in exactly reproducing the density profile while splitting and merging particles. As a consequence, pressure shocks and thus oscillating particles are visible during this process. As long as the particle movement dominates the velocity field these shocks are hardly visible, but as soon as the fluid comes to rest these artifacts are clearly apparent. Other techniques to speed up the computations include hybrid simulation models that couple two-dimensional with three-dimensional

fluid simulations, for example a 2D shallow water simulation with a 3D free surface fluid simulation model as presented in [Thürey et al., 2006].

3) ***Incompressibility condition***
In the standard SPH model (e.g. [Monaghan, 1992; Müller et al., 2003; Müller et al., 2005b]), the pressures are computed using a soft EOS resulting in undesired compression artifacts. Although incompressibility can be enforced, it represents the most expensive part of the whole simulation process and thus renders particle methods less attractive for high quality and photorealistic animations of water. In the literature, two different strategies have been pursued to model incompressibility. First, the weakly compressible SPH (WCSPH) method has been used where pressure is modeled using a stiff equation of state (e.g. [Monaghan, 2005; Becker and Teschner, 2007; Becker et al., 2009]), and second, incompressibility has been achieved by solving a pressure Poisson equation (ISPH methods, e.g. [Cummins and Rudman, 1999; Shao, 2006; Hu and Adams, 2007]) similar to Eulerian fluid solvers. Although both methods satisfy incompressibility, the computational expenses of simulating high resolution fluid animations are too large for practical use. The drawback of WCSPH is the severe time step restriction since the stiffness of the fluid typically dominates the Courant-Friedrichs-Levy (CFL) condition. Thus the computational cost increases with decreasing compressibility. Although ISPH allows larger time steps, the computational cost per physics step is immensely higher. Furthermore, the complexity to formulate and solve the equation system on unstructured particle configurations represents a major problem and is computationally expensive.

4) ***Interfaces of multiple fluids***
In standard SPH, particles have a spatial distance (smoothing length) over which their properties are smoothed by a kernel function [Monaghan, 1992]. Problems arise at interfaces of multiple fluids with density contrast, since the rest densities and masses of neighboring particles vary within the smoothing length. As a result, the smoothed quantities of a particle show falsified values and the physical behavior is severely affected, especially if density ratios become large [Hoover, 1998; Agertz et al., 2006]. The undesirable effects reach from falsified densities and pressures to spurious and unnatural interface tensions, degrading the visual result tremendously. With increasing density ratios, one has additionally to cope with severe numerical instabilities which cannot be overcome by decreasing the time step of the simulation alone [Colagrossi and Landrini, 2002].

5) ***Complex fluid-solid interactions***
In current simulation systems, fluids and solids are often discretized by differ-

ent models thus restricting the number of interaction effects which can be simulated (e.g. [Terzopoulos et al., 1989; Cani and Desbrun, 1997; Müller et al., 2004]). To facilitate the interaction between fluids and solids, it is highly desirable to have a single simulation method that can handle different types of materials [Müller et al., 2004; Keiser et al., 2005] and that is able to combine several properties and effects in a single model. Examples of such properties are melting and solidification even while being surrounded by other liquids, the distinction of multiple objects which are touching as long as no melting is involved, the ability to merge multiple close objects into one when melting is involved, and to split objects as a result of phase changes. Moreover, coarsely sampled and even coplanar particle configurations should be handled without further treatment since they often result from phase change processes.

6) ***Reconstructing smooth surfaces***
Another challenge in particle animation methods is to reconstruct visual appealing surfaces from the particles. Applications which have to run in real-time or at interactive frame rates suffer from blobby surfaces and smoothed surface details since the number of simulated particles is limited [Müller et al., 2003]. When using a point splatting approach as rendering technique, undersampled geometries show artifacts at the silhouette and blur due to large splat radii (e.g. [Guennebaud et al., 2004; Solenthaler et al., 2007b]). Hence it would be desirable to improve the visual quality of low resolution particle simulations that can be simulated at interactive frame rates.

Unfortunately, with high-resolution simulations bumpy surfaces related to the particle distribution are still visible. Although the color-field presented in [Müller et al., 2003] improves the trivial approach of blobbies [Blinn, 1982], flat surfaces cannot be achieved due to the irregular particle distribution. [Zhu and Bridson, 2005] proposed an approach involving the center of mass of neighboring particles resulting in smooth surfaces. Normal vectors are not involved in their reconstruction technique, avoiding the problem of incorrect particle normals in splashes and drops which are typically represented by a few particles only. Unfortunately, their approach leads to significant artifacts in concave regions which they suggest to remove in a post-processing step. However, difficulties exist in distinguishing between artifacts and real surface, thus erroneous surface parts cannot be removed completely.

7) *Boundary deficiency*
The standard density summation equation of SPH has problems at the fluid boundaries since the neighborhoods of particles close to the boundary are not completely filled. As a result, densities are underestimated in such situations thus causing errors in the resulting pressure and force fields. Although the

density convergence equation, where densities are initially set and evolved over time, does not suffer from the deficiency problem, one has to cope with stability issues due to the accumulation of density integration errors. Hence the use of the convergence equation is typically omitted in graphics applications since large time steps and long-term simulations exacerbate the problems. In the literature, the deficiency problem is typically addressed by using ghost particles along the domain boundaries, for example by mirroring the fluid particles at the walls (e.g. [Morris et al., 1997]). However, this is only applicable if the boundary is simple, and has the disadvantage of increased computational cost since ghost particles have to be included in the physics computation. Another solution is the use of the corrected SPH (CSPH) method where adapted kernel functions are used that compensate for the missing neighborhood [Bonet and Kulasegaram, 2002; Becker et al., 2009].

8) *Viscous fluids*
Liquids simulated with SPH often appear to be too viscous compared to liquids in the real world. The main reason for that is the artificial viscosity force that has to be added in order to stabilize the numerical algorithm ([Lucy, 1977; Monaghan, 1992]). Viscous forces are acting between each particle and its neighbors inside the support radius, accelerating the particles in the direction of the relative speed of its environment. The size of the support radius is chosen so large such that it includes around 30-40 particles. When increasing the particle resolution, the support domain can be reduced hence covering a smaller area of the simulation domain. This reduces the damping problem of the artificial viscosity. However, the behavior does also depend on the characteristics of the viscosity formulation. Different techniques have been proposed (e.g. [Hernquist and Katz, 1989; Balsara, 1995; Morris and Monaghan, 1997]), but further studies and comparisons need to be done in order to evaluate the formulation that meets the demands of graphics applications best.

Another defect of SPH that visually results in a relatively strong viscous behavior is that pressure forces compel the particles to arrange in a stable lattice structure [Lombardi et al., 1999]. This crystallization prevents particles to move since they have to break open the stable particle configuration at first. As a result, small turbulences and buoyancy effects are damped.

1.4 Contributions

In this thesis, we focus on the elimination of some of the major problems of particle-based simulations, in particular we address the issues 3) - 6) previously

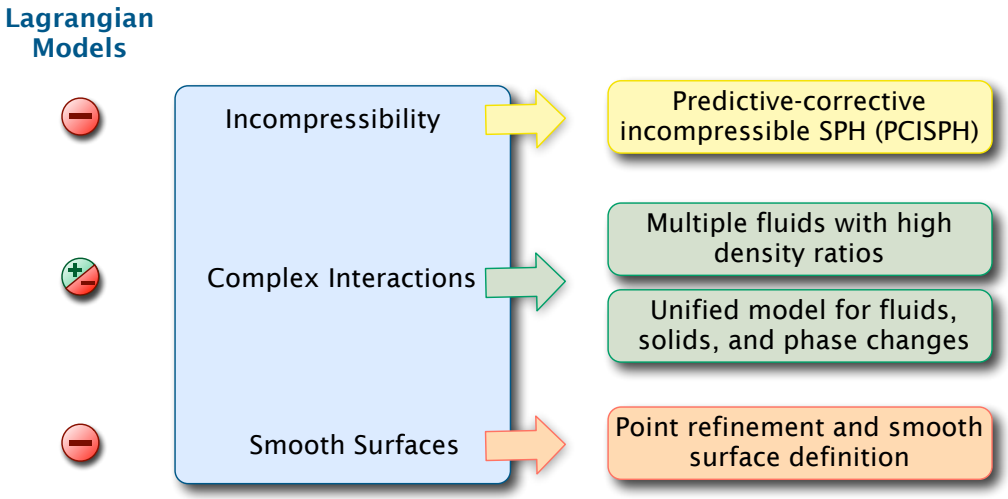


Figure 1.4: *In this thesis, we focus on improving some of the major problems of particle-based simulations. We present a new SPH method that can efficiently handle incompressibility, we propose a modified SPH model that eliminates the artifacts at the interface of multiple fluids with high density ratios, we demonstrate a unified particle model to simulate fluids, solids, and phase changes, and we show a surface point refinement and reconstruction method improving the smoothness of the fluid surface.*

described in Section 1.3. We introduce new methods to solve these problems while bearing in mind the resulting efficiency of the simulation. First, we focus on the major disadvantage of particle systems, which is the efficient solution of the incompressibility condition. Second, we focus on complex interaction effects. Although one strength of particles is the ease to handle interaction effects with complex boundaries, there still exist severe problems when simulating specific effects like multiple fluids with high density ratios or phase change processes. And third, we enhance the smoothness of the fluid surfaces by sophisticated point refinement and surface reconstruction techniques. The illustration in Figure 1.4 gives an overview over the contribution of this thesis. In the following, we summarize our solutions:

1) ***Incompressibility condition:***
Predictive-corrective incompressible SPH method
We present a novel, incompressible fluid simulation method based on the Lagrangian SPH model. In our method, incompressibility is enforced by using a prediction-correction scheme to determine the particle pressures. For this, the information about density fluctuations is actively propagated through the fluid and pressure values are updated until the targeted density is satisfied. With this

approach, we avoid the computational expenses of solving a pressure Poisson equation, while still being able to use large time steps in the simulation. The achieved results show that our predictive-corrective incompressible SPH (PCISPH) method clearly outperforms the commonly used weakly compressible SPH (WCSPH) model by more than one order of magnitude while the computations are in good agreement with the WCSPH results.

2) ***Interfaces of multiple fluids:***
Adapted SPH for fluids with density ratios
To overcome the problems at interfaces present in the standard SPH method, we propose a modified SPH formulation that can handle density discontinuities at interfaces between multiple fluids correctly without increasing the computational costs compared to standard SPH. The basic idea is to replace the density computation in SPH by a measure of particle densities and consequently derive new formulations for pressure and viscous forces. These modifications of the SPH formulation corrects for density problems, spurious and unphysical interface tension, and instabilities otherwise present at high density contrast interfaces. The elimination of these artifacts enables an animator to fully control the physical behavior of multiple fluids, including the selection of the desired amount of interface tension according to the simulation problem at hand. Our modifications do not increase the overall computational cost compared to the standard SPH model.

3) ***Complex fluid-solid interactions:***
A unified particle model to simulate fluids, solids, and phase changes
We present a new method for the simulation of melting and solidification in a unified particle model. Our technique uses the SPH method for the simulation of liquids, deformable as well as rigid objects, which eliminates the need to define an interface for coupling different model representations. Using this approach, it is possible to simulate fluids and solids by only changing the attribute values of the underlying particles. We significantly changed a prior elastic particle method to achieve a flexible model for melting and solidification. By using an SPH approach and considering a new definition of a local reference shape, the simulation of merging and splitting of different objects, as may be caused by phase change processes, is made possible. In order to keep the system stable even in regions represented by a sparse set of particles we use a special kernel function for solidification processes. The results demonstrate new interaction effects regarding the melting and solidification of material, even while being surrounded by liquids.

4) ***Reconstructing smooth surfaces***
A particle refinement and surface reconstructing method improving the

surface quality
A novel point refinement method is presented for irregularly sampled, dynamic points coming from a particle-based fluid simulation. Our interpolation algorithm can handle complex geometries including splashes, and at the same time preserves features like edges. Point collisions are avoided resulting in a nearly uniform sampling facilitating surface reconstruction techniques. No point preprocessing is necessary, and point neighborhoods are dynamically updated reducing computation and memory costs. We show that our algorithm can efficiently detect and refine the surface points of a fluid and we demonstrate the improvement of rendering quality and applicability to real-time simulations. Additionally, we propose a surface reconstruction technique producing smooth surfaces without requiring high quality point normals. To avoid artifacts in concave regions, our reconstruction is based on considering the movement of the center of mass which reduces these rendering errors.

1.5 Dissertation Overview

This dissertation first gives a brief description of the SPH formalism in Chapter 2. Based on the equations of motion of a fluid, the basic formalism of SPH is derived and the resulting SPH equations for the density, pressure, and forces are summarized. The animation loop of the standard SPH algorithm is described in the following. Further, the compressibility problem due to the equation of state is discussed, and the weakly compressible SPH (WCSPH) model is introduced. We discuss the time stepping and present measurements that illustrate the time step restriction of stiff fluid systems.

Chapter 3 introduces our new incompressible SPH method based on a prediction-correction scheme. Our results show that our model combines the advantages of both WCSPH and incompressible SPH (ISPH) in one model, namely low computational cost per physics update and large time steps. Performance results are presented which show that our new method outperforms the commonly used WCSPH by more than an order of magnitude, while the physical behavior is in good agreement with the WCSPH results.

In Chapter 4 we focus on the interface problems of multiple fluids with density contrast. We illustrate the smoothing problem of SPH and show how the falsified quantities lead to spurious interface tension and instability problems. We demonstrate how the equations of a standard SPH implementation can be adapted to avoid physical artifacts and stability problems when simulating multiple fluids with high density ratios. We show that our modifications do not increase the over-

all computation cost of SPH while improving the physical behavior significantly.

Chapter 5 presents a flexible, unified model based on SPH to simulate fluids, deformable bodies, and rigid bodies represented by particles. We discuss how melting and solidification effects can be modeled, and how merging and splitting due to phase changes can be stably achieved. Various animation effects are shown, including complex interaction between fluids, elastic objects, and solid objects, as well as phase changes between the states.

In Chapter 6 we describe how the visual quality of low-resolution fluid surfaces can be improved by upsampling the surface points using our spherical interpolation method. It is explained how computation and memory costs can be reduced by using efficient algorithms for neighborhood update and collision avoidance. Further, we show how smooth surfaces can be extracted from fluid point data without the need for high quality normals. Artifacts in concave regions are successfully avoided by investigating the movement of the center of mass.

Each Chapter 3-6 includes a thorough discussion about the specific problem as well as relevant related work. Results and discussions can also be found in the particular chapters.

We conclude the dissertation in Chapter 7 with a summary and directions for future work.

CHAPTER 2

SMOOTHED PARTICLE HYDRODYNAMICS

2.1 Equations of Motion

In SPH, the physical forces acting on the particles are derived from the Navier-Stokes (NS) equations. The conservation of momentum and the conservation of mass are written as

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \rho \mathbf{g} + \mu \nabla^2 \mathbf{v} \tag{2.1}$$

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}, \tag{2.2}$$

where ρ is the density, $\mathbf{v}$ the velocity, p the pressure, $\mathbf{g}$ an external body force, such as gravity, and μ the viscosity of the fluid. Note that $\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + \mathbf{v} \cdot \nabla q$ is the material derivative denoting the change of a quantity q over time. Since in the Lagrangian viewpoint the quantity moves with the fluid (i.e. with the particles), the material derivative equals the partial time derivative, meaning that $\frac{Dq}{Dt} = \frac{\partial q}{\partial t}$ and that the convective term $\mathbf{v} \cdot \nabla q$ is zero.

Equation 2.1 is simply Newton's second law $\mathbf{F} = m\mathbf{a}$ which can easily be seen by multiplying both sides of the equation by the volume V_i of a particle i

$$m_i \frac{D\mathbf{v}_i}{Dt} = -V_i \nabla p_i + m_i \mathbf{g} + V_i \mu \nabla^2 \mathbf{v}_i. \tag{2.3}$$

On the right hand side of Equation 2.3 there are three forces, the pressure force $-V_i \nabla p_i$, the external force $m_i \mathbf{g}$, and the viscous force $V_i \mu \nabla^2 \mathbf{v}_i$

$$m_i \frac{D\mathbf{v}_i}{Dt} = \mathbf{F}_i^{pressure} + \mathbf{F}_i^{external} + \mathbf{F}_i^{viscosity}. \tag{2.4}$$

For the acceleration $\mathbf{a}$ of a particle i we then get

$$\mathbf{a}_i = \frac{D\mathbf{v}_i}{Dt} = \frac{\mathbf{F}_i}{m_i}, \tag{2.5}$$

where $\mathbf{F}_i$ is the sum of all forces acting on i. To determine the acceleration $\mathbf{a}_i$ of a particle i in the Lagrangian simulation environment, the density ρ_i, the pressure p_i, and the total force $\mathbf{F}_i$ have to be derived. For this, the SPH approximations described in Section 2.3 can be applied.

Equation 2.2 is called the mass-conservation or continuity equation. For an incompressible fluid the density does not change, corresponding to $\frac{D\rho}{Dt} = 0$. In the Lagrangian formulation, this equation is typically not satisfied due to the immense computational costs. Rather, the liquid is typically modeled as a compressible fluid which reduces the costs but degrades the visual quality of the animated liquid. This is discussed in detail in Section 2.5 and Chapter 3.

2.2 Basic Formalism

SPH, originally developed for the simulation of astrophysical problems [Gingold and Monaghan, 1977; Lucy, 1977], uses the concept of integral representation of a function or field variable $A(\mathbf{x})$

$$A(\mathbf{x}) = \int_\Omega A(\mathbf{x}')\delta(\mathbf{x} - \mathbf{x}')d\mathbf{x}' \tag{2.6}$$

where Ω is the domain, $d\mathbf{x}'$ the differential volume element, and $\delta(\mathbf{x} - \mathbf{x}')$ is the Dirac delta function. If the Dirac delta function is replaced by a smoothing kernel function W with support radius or smoothing length h, the integral representation of $A(\mathbf{x})$ is given by

$$A(\mathbf{x}) = \int_\Omega A(\mathbf{x}')W(\mathbf{x} - \mathbf{x}', h)d\mathbf{x}'. \tag{2.7}$$

Note that as long as W is not the Dirac delta function, the integral representation in Equation 2.7 is only an approximation. The kernel function W should satisfy a number of conditions. First, the normalization condition states that the kernel function has to be normalized according to

$$\int W(\mathbf{x} - \mathbf{x}', h)d\mathbf{x}' = 1. \tag{2.8}$$

Second, the Delta function property has to be satisfied

$$\lim_{h \to 0} W(\mathbf{x} - \mathbf{x}', h) = \delta(\mathbf{x} - \mathbf{x}'), \tag{2.9}$$

and third, the compact condition defines the non-zero area of the smoothing function

$$W(\mathbf{x} - \mathbf{x}', h) = 0 \quad \text{when } |\mathbf{x} - \mathbf{x}'| > h.$$

The smoothing kernel is most often approximated by spline kernels with finite support [Monaghan, 1992; Müller et al., 2003].

The integral in Equation 2.7 is discretized onto a finite set of interpolation points, the fluid particles, by replacing the integral by a summation and the differential volume element $d\mathbf{x}'$ by the volume V which is the mass m divided by the density ρ (Figure 2.1)

$$A(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{x} - \mathbf{x_j}, h). \tag{2.10}$$

Gradient and Laplacian values can be easily calculated by taking the first and second derivative of the kernel, respectively

$$\nabla A(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} A_j \nabla W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.11}$$

$$\nabla^2 A(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W(\mathbf{x} - \mathbf{x}_j, h). \tag{2.12}$$

2.3 Governing Equations

2.3.1 Density

In the literature, there are two different equations used to compute the density. Typically, the *standard density summation* is implemented, where the SPH interpolation is applied to the density field resulting in

$$\rho_i = \sum_j m_j W(\mathbf{x}_{ij}, h). \tag{2.13}$$

With this equation, densities of particles close to the boundary are underestimated due to empty parts in the neighborhood, thus causing errors in the resulting pressure and force fields.

Alternatively to the standard density summation, the density can be evolved over time according to the *continuity equation* (convergence equation) [Monaghan, 1994]. With this formulation, particle densities are initially set and evolved

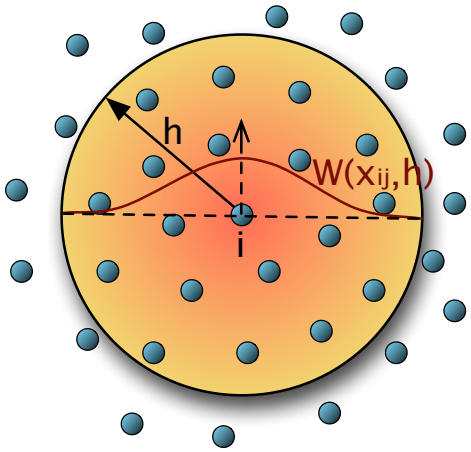


Figure 2.1: *In SPH, the particle quantities are smoothed over a neighborhood of size h. To evaluate a quantity at a particle's location, the weighted contributions of the neighbors are summed up, while close neighbors have a larger weight than neighbors farther away.*

during the simulation by computing the rate of change of the density of each particle i by

$$\frac{d\rho_i}{dt} = \sum_j m_j(\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla W(\mathbf{x}_{ij}, h). \tag{2.14}$$

Although the convergence equation does not suffer from underestimated densities at free surfaces due to the particle deficiency at such locations, it does not produce stable results for long-term simulations. This is due to severe density integration errors, especially when using large time steps and low-order time integration schemes which is important for the targeted type of graphics applications. Therefore, the models used throughout this thesis all implement the standard density summation equation.

2.3.2 Pressure and Pressure Forces

The pressure p_i of a particle is then derived from the equation of state (EOS) according to [Batchelor, 1967]

$$p_i = \frac{k\rho_0}{\gamma}((\frac{\rho_i}{\rho_0})^\gamma - 1), \tag{2.15}$$

where k controls the stiffness of the fluid, ρ_0 is the reference density, and γ is usually set to 7. Note that when using a γ of 1, Equation 2.15 corresponds to the

pressure formulation in [Desbrun and Cani, 1996], which is

$$p_i = k\rho_0((\frac{\rho_i}{\rho_0}) - 1) = k(\rho_i - \rho_0). \tag{2.16}$$

In WCSPH, k is chosen so that the speed of sound is large enough to keep the density fluctuations small ($\sim 1\%$) (see Section 2.5.1). Note that the CFL condition requires smaller time steps for stiffer fluids which increases the overall computation cost tremendously when simulating water (see Section 2.5.3).

In [Müller et al., 2003], the pressure force field is then computed by

$$\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i}\sum_j \frac{m_j}{\rho_j}\frac{p_i + p_j}{2}\,\nabla W(\mathbf{x}_{ij}, h). \tag{2.17}$$

Alternatively to Equation 2.17, the pressure force equation according to [Monaghan, 1992] can be used:

$$\mathbf{F}_i^{pressure} = -\sum_j m_i m_j(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2})\nabla W(\mathbf{x}_{ij}, h). \tag{2.18}$$

We compare these two equations in the context of multiple fluids in Chapter 4. If not stated otherwise we use Müller's Equation 2.17.

2.3.3 Viscous Forces

Viscous forces are mainly used to stabilize a particle system. Several formulations have been presented for different applications (e.g. [Hernquist and Katz, 1989; Balsara, 1995; Morris and Monaghan, 1997]). If not stated otherwise we use the following equation from [Müller et al., 2005b]

$$\mathbf{F}_i^{viscosity} = \frac{m_i}{\rho_i}\sum_j \frac{\mu_i + \mu_j}{2}\frac{m_j}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i)\,\nabla^2 W(\mathbf{r}_{ij}, h), \tag{2.19}$$

where μ is the viscosity constant of a particle defining the strength of viscosity.

2.3.4 Smoothing Kernels

If not stated otherwise, we use the following polynomial kernel presented in [Müller et al., 2003] which can be efficiently computed

$$W_{poly6}(\mathbf{x}_{ij}, h) = \frac{315}{64\pi h^9}\begin{cases} (h^2 - x_{ij}^2)^3 & 0 \le x_{ij} \le h \\ 0 & \text{otherwise,} \end{cases} \tag{2.20}$$

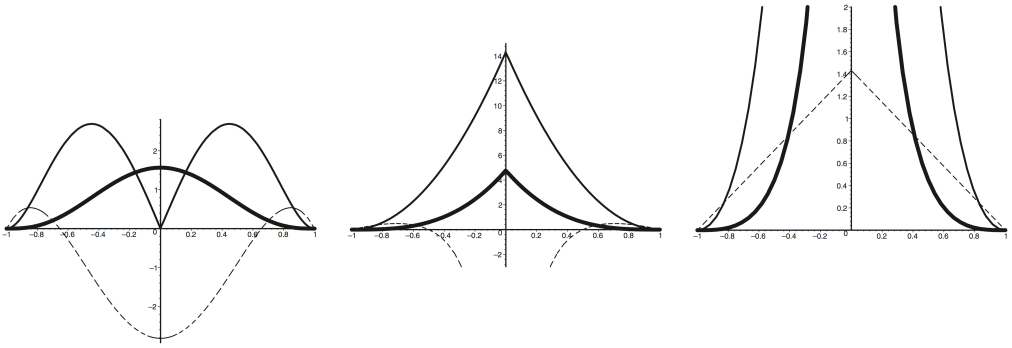


Figure 2.2: *The smoothing kernels W_{poly6}, W_{spiky}, and $W_{viscous}$ presented in [Müller et al., 2003]. The thick, thin, and dotted lines show the kernels, their gradients, and Laplacians, respectively. Image courtesy of Matthias Müller, NVIDIA.*

where $x_{ij} = ||\mathbf{x}_{ij}||$. However, if this kernel is used for the computation of the pressure forces particles tend to cluster if they get close to each other. This happens because the gradient approaches zero at the center and thus repulsion forces vanish. For pressure computations, therefore, a spiky kernel is used

$$W_{spiky}(\mathbf{x}_{ij}, h) = \frac{15}{\pi h^6} \begin{cases} (h - x_{ij})^3 & 0 \leq x_{ij} \leq h \\ 0 & \text{otherwise.} \end{cases} \tag{2.21}$$

The viscous forces are computed using a third kernel

$$W_{viscous}(\mathbf{x}_{ij}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{x_{ij}^3}{2h^3} + \frac{x_{ij}^2}{h^2} + \frac{h}{2x_{ij}} - 1 & 0 \leq x_{ij} \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{2.22}$$

whose Laplacian is positive everywhere. This is an important property since otherwise the resulting viscous forces acting between close particles increase the particles' relative velocities and thus decrease the stability of the simulation.

2.3.5 Time Integration

Generally, time integration schemes are classified either as explicit or implicit. Explicit methods calculate the state of a system at time $t + 1$ from the state of the system at the current time t, while implicit methods find a solution by solving an equation involving both the current state of the system and the later one. With implicit methods, stability is maintained for large time steps, but large computing time per integration step is required. Although explicit methods require small time steps for stability, they allow fast computation per integration step, which

is essential if frequent updates are required. In SPH, explicit time integration schemes are typically used, while first- and second-order methods are favored due to their fast execution. In most cases, the Leap-Frog method is applied because of its simplicity and second-order accuracy.

After having calculated all particle accelerations $\mathbf{a}_i$, the velocities and positions of the particles are forwarded in time using the Leap-Frog time integration scheme. The velocity and position of the next simulation step are given by

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F_i}}{m_i} \tag{2.23}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+1), \tag{2.24}$$

where Δt is the time step size which can be determined by the CFL condition (see Section 2.5). Its simplicity and the second-order accuracy make this scheme attractive for efficient animations. Note that in the Leapfrog scheme positions and velocities are off by half a time step.

2.4 Animation Loop

One simulation step includes the computation of the equations discussed above. In our current implementation, we use four passes over all particles, summarized in Algorithm 1.

First, for each particle, the neighboring particles within the range h have to be found. Several search data structures have been proposed differing in computational complexity and memory consumption. Non-hierarchical data structures such as grids can be built and updated efficiently, while hierarchical data structures like octrees and kd-trees often allow faster queries than grids. However, the performance depends largely on the particle number used in the simulation [Keiser, 2006]. In this thesis we use a uniform 3D grid with cells of size h. Thus, for each particle, 27 cells have to checked to find the neighbors. The support radius h is typically chosen so that the average number of neighbors of a particle is around 30-40.

After having determined the neighbors of each particle, the densities and pressures can be computed with Equations 2.13 and 2.15. Since each particle has to know the quantities of its neighbors to compute the forces, an additional pass is needed to compute the pressure force (Equation 2.17), viscous force (Equation 2.19), and external forces (for example tension forces which are discussed in Chapter 4).

Then, the forces are used to integrate the velocities and positions forward in time (Equations 2.23 and 2.24), and the new particle positions are checked for collisions with the domain boundary.

These steps are repeated as long as the animation is running.

Algorithm 1 SPH / WCSPH

```
1  while animating do
2    for all i do
3      find neighborhoods N_i(t)
4    end for
5    for all i do
6      compute density ρ_i(t)
7      compute pressure p_i(t)
8    end for
9    for all i do
10     compute forces F^{p,v,g,ext}(t)
11   end for
12   for all i do
13     compute new velocity v_i(t + 1)
14     compute new position x_i(t + 1)
15     collision handling
16   end for
17 end while
```

2.5 Compressibility Analysis

2.5.1 Weakly Compressible SPH

In reality, water is slightly compressible. Sound waves are traveling through the medium, meaning that volume and thus density gets perturbed. However, this volume change can hardly be noticed, even in the ocean at 4000m depth where pressures are $4 \cdot 10^7 Pa$, there is only a 1.8% decrease in volume. At 10m depth, the volume decrease is only 0.0045%. These tiny perturbations in the volume have so a small effect on how fluids move at macroscopic level that they are practically irrelevant for animation. For that reason, water is often treated as incompressible which means that the volume stays constant. In finite difference methods, this incompressibility constraint is enforced by solving a pressure Poisson equation to determine the appropriate pressure values. Although the same concept can be applied to particles, the equation system gets more complex and thus computationally more expensive to solve. Therefore, SPH typically approximates water by an artificial fluid which is slightly compressible. Analogously to standard SPH, the weakly compressible SPH (WCSPH) method relates the density and the pressure using the Tait equation (Equation 2.15). The difference is that WCSPH uses such a large value for k so that the speed of sound $c_s = \sqrt{k}$ is large enough to

keep density fluctuations $\frac{|\delta\rho|}{\rho_0}$ small. $\delta\rho$ corresponds to the deviation from the rest density ρ_0. According to [Monaghan, 2005],

$$\frac{|\delta\rho|}{\rho_0} \sim \frac{|\mathbf{v}|^2}{c_s^2}, \tag{2.25}$$

where $\mathbf{v}$ is the maximum speed of the fluid and c_s is the speed of sound. Thus, to ensure that $\frac{|\delta\rho|}{\rho_0} \sim \eta$, k can be determined by

$$k = c_s^2 = \frac{|\mathbf{v}|^2}{\eta}. \tag{2.26}$$

Typically, η is set to 0.01, thus allowing density varitations of 1%. To enforce the condition given by Equation 2.26, an estimate of the maximum speed is required. This is often quite difficult to do without running the simulation in advance, since the fluid might interact with complex domain boundaries and solid objects. Even worse, for many applications it is not sufficient to run the animation in advance since there might be unforeseen external forces or sudden impacts of solid bodies which influences the maximum speed significantly. Consequently, an animator cannot get around extensive testing and parameter tuning to find the appropriate k which enforces $\eta \leq 1\%$.

2.5.2 Speed of Sound

To demonstrate the influence of k more clearly, we have set up a test scene consisting of a tube of length 1m. The tube is closed to all sides, and the particles fill the tube completely (Figure 2.3). Gravitation is turned off in this example, and the particles are initially at rest. In the first iteration, a pressure wave is initiated on the left side of the tube. For different values for k, we have measured the time when the wave reaches the particles on the right side. The measurements are summarized in Table 2.1. In the real world, the speed of sound in water is $1497\frac{m}{s} \approx 1500\frac{m}{s}$ at $25°$ Celsius. This means that a pressure wave, initiated on the left side of the tube and propagating through the fluid, reaches the right side of the tube after $t = \frac{1}{1500}s \approx 6 \cdot 10^{-4}s$. To reach the same time in the simulation, a k between 10^6 and 10^7 has to be chosen. Note that in the graphics literature k is often set to 1000, which for our example means that information is propagated approximately 68 times slower than in the real world (Table 2.1). Obviously, this negatively affects the fluid behavior since splashes and waves emerge delayed. Another defect is the "breathing" behavior which can be observed with high compressibility since particles get too strongly compressed and pressure waves get reflected. This energy loss further reduces or prevents waves and splashes from evolving. With a k of (at least) 10^6 that is needed to achieve the correct wave

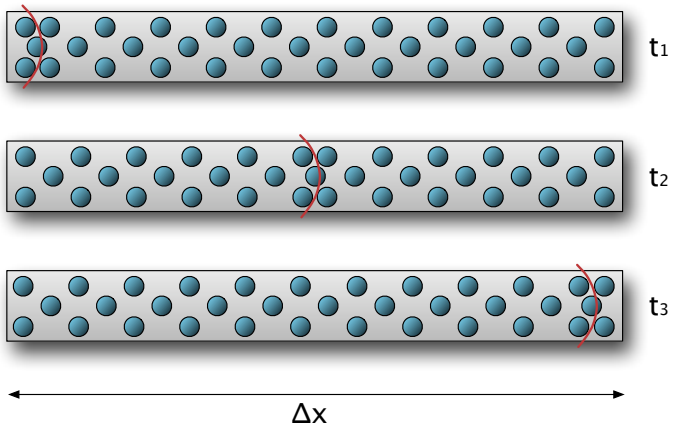


Figure 2.3: *Our test scene is a tube of length $\Delta x = 1m$ completely filled with particles. Gravitation is turned off in this scene, and the particles are initially in rest. A pressure wave (indicated in red) is initiated on the left side of the tube at time t_1. The wave propagates through the fluid and reaches the right side of the tube at time t_3. For different stiffness values k, we measure the time when the compression wave reaches the other end of the tube.*

k	10^1	10^2	10^3	10^4	10^5	10^6	10^7
$t[s]$	0.5988	0.1734	0.0413	0.0118	0.0034	0.0011	0.0003

Table 2.1: *Information propagation measurements with different values for k.*

speed in our test scene, and a support radius h of 0.02, the time step has to be set to $8.0 \cdot 10^{-6}s$ according to the CFL condition given by Equation 2.27. This means that 125'000 simulation steps have to be executed in order to simulate $1s$ of real time.

To verify that the size of the time step does *not* influence the wave speed, we have fixed the stiffness of the fluid to $k = 100$ and have again measured the propagation time using different time step sizes Δt. Table 2.2 shows that the propagation time, as expected, does not change with smaller Δt (the small differences in time can be explained by time integration and measurement inaccuracies).

2.5.3 CFL Condition

The numerical integration can be carried out by standard methods, like the Leap-frog scheme which is used in the implementation of this thesis, with a time-step

Δt	$6 \cdot 10^{-4}$	$6 \cdot 10^{-5}$	$6 \cdot 10^{-6}$	$6 \cdot 10^{-7}$
$t[s]$	0.1734	0.1712	0.1727	0.1802

Table 2.2: *Information propagation measurements with different time step sizes Δt for k set to* 100*.*

control involving a CFL condition. The condition ensures that an information which propagates with velocity $\mathbf{v}$ does not leave out some grid points in a grid with cell size Δx, giving the condition $\frac{\mathbf{v}\Delta t}{\Delta x} \leq 1$. Otherwise such leaps will result in a lack of information at these points and will give rise to instability. Applied to particles, the CFL condition requires that no particle is left out, meaning that

$$\Delta t_c \leq \alpha \frac{h}{c}, \tag{2.27}$$

where c is the speed of sound and α is approximately 0.4. When simulating highly viscous fluids, the condition has to be extended to account for the viscous terms [Monaghan, 1992]. But for low viscous fluids like water the term incorporating the speed of sound dominates the viscous terms and thus the viscous terms can be neglected. When dealing with large forces the condition has to be extended with the term Δt_f considering the force per unit mass $\mathbf{f}$ as described in [Monaghan, 1992]

$$\Delta t_f = 0.25 \cdot \min_i (\frac{h}{|\mathbf{f}_i|})). \tag{2.28}$$

Then, the resulting time step Δt is given by

$$\Delta t = \min(\Delta t_c, \Delta t_f). \tag{2.29}$$

In WCSPH, where density fluctuations are held below 1%, Δt_c usually dominates the condition as k and c_s, respectively, have to be chosen sufficiently large. As the equation indicates, the time step size has to be decreased with increasing stiffness, i.e. with increasing c_s, to get a stable simulation.

CHAPTER 3

PREDICTIVE-CORRECTIVE DENSITY

3.1 Incompressible SPH

Enforcing incompressibility in fully particle-based fluid simulations represents the most expensive part of the whole simulation process and thus renders particle methods less attractive for high quality and photorealistic water animations. In the context of Smoothed Particle Hydrodynamics (SPH), two different strategies have been pursued to model incompressibility. First, the weakly compressible SPH (WCSPH) method has been used where pressure is modeled using a stiff equation of state (EOS), and second, incompressibility has been achieved by solving a pressure Poisson equation. Although both methods satisfy incompressibility, the computational expenses of simulating high resolution fluid animations are too large for practical use.

In the standard SPH and WCSPH model the particle pressures are determined by an EOS. The characteristics of this equation and the stiffness parameter determine the speed of the acoustic waves in a medium. The EOS-based SPH with low stiffness according to [Desbrun and Cani, 1996] was used in a series of papers to simulate water [Müller et al., 2003; Adams et al., 2007], multiple fluids [Müller et al., 2005b; Solenthaler and Pajarola, 2008], fluid-solid coupling [Müller et al., 2004; Lenaerts et al., 2008; Becker et al., 2009], melting solids [Müller et al., 2004; Keiser et al., 2005; Solenthaler et al., 2007a], and fluid control [Thürey

et al., 2006]. In contrast to the standard SPH formulation, WCSPH uses a stiff EOS [Monaghan, 2005; Becker and Teschner, 2007; Becker et al., 2009] resulting in acoustic waves traveling closer to their real speed through the medium. Typically, the stiffness value is chosen so large that the density fluctuations do not exceed 1%. The required stiffness value to achieve this, however, is difficult or even impossible to determine before running the simulation. Consequently, an animator cannot get around extensive testing and parameter tuning. Another drawback is that WCSPH imposes a severe time step restriction as the stiffness of the fluid usually dominates the Courant-Friedrichs-Levy (CFL) condition. Thus the computational cost increases with decreasing compressibility – since higher stiffness requires smaller time steps, making it infeasible to simulate high resolution fluids within reasonable time.

Rather than simulating acoustic waves, incompressibility in Lagrangian methods can be enforced by solving a pressure projection similar to Eulerian methods (e.g. [Enright et al., 2002]). These incompressible SPH (ISPH) methods first integrate the velocity field in time without enforcing incompressibility. Then, either the intermediate velocity field [Cummins and Rudman, 1999], the resulting variation in particle density [Shao, 2006], or both [Liu et al., 2005; Hu and Adams, 2007; Losasso et al., 2008] are projected onto a divergence-free space to satisfy incompressibility through a pressure Poisson equation. With these ISPH methods density fluctuations of 1% to 3% have been reported. A problem with these methods, however, is the complexity to formulate and solve the equation system on unstructured particle configurations. Although ISPH allows larger time steps than WCSPH, the computational cost per physics step is much higher. A Poisson solver was also used in [Premoze et al., 2003] for the particle method Moving-Particle Semi-Implicit (MPS), increasing the cost per physics time step enormously. In contrast to the fully Lagrangian models, [Zhu and Bridson, 2005] propose to use an auxiliary background grid to simplify the equation system to a sparse set of linear equations which can be efficiently solved. A similar hybrid solver for vorticity confinement is presented in [Selle et al., 2005]. In [Losasso et al., 2008], a two-way coupled level set method with an SPH solver is introduced to simulate dense and diffuse water volumes. They demonstrate how to enforce incompressibility and target the particle number density with a single Poisson solve.

In this thesis, we propose a novel, fully Lagrangian, incompressible SPH method featuring the advantages of both WCSPH and ISPH in one model, namely low computational cost per physics update and large time steps (Figure 3.1). Our method makes use of a prediction-correction scheme which propagates the estimated density values through the fluid and updates the pressures in such a way that incompressibility is achieved. The propagation stops as soon as a previously user-defined density variation limit is reached for each individual particle. We will show in this thesis that our new *predictive-corrective incompressible* SPH

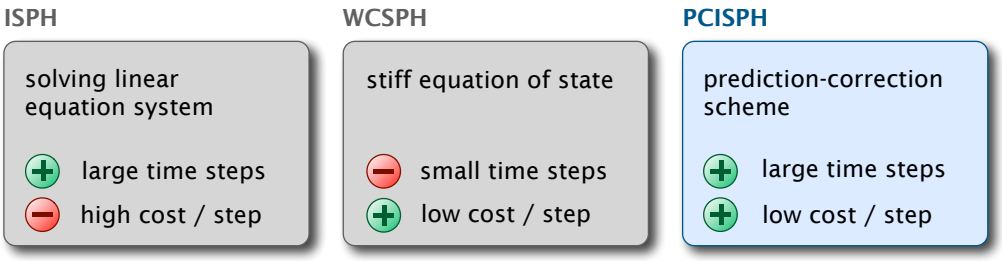


Figure 3.1: *Comparison of different incompressible SPH methods. Our predictive-corrective incompressible SPH (PCISPH) method features the advantages of both ISPH and WCSPH: large time steps and low computational cost per physics update.*

(PCISPH) method outperforms WCSPH by more than an order of magnitude while the computations are in good agreement with the WCSPH results. The efficiency of our method enables an animator to produce high-resolution fluid animations within reasonable time without compressibility artifacts.

3.2 PCISPH Model

3.2.1 PCISPH Algorithm

To avoid the time step restriction of WCSPH we propose to use a prediction-correction scheme based on the SPH algorithm (PCISPH). In our method, the velocities and positions are temporarily forwarded in time and the new particle densities are estimated. Then, for each particle, the predicted variation from the reference density is computed and used to update the pressure values, which in turn enter the recomputation of the pressure forces. Similar to a Jacobi iteration for linear systems, this process is iterated until it converges, i.e. until all particle density fluctuations are smaller than a user-defined threshold η (for example 1%). Note that this is a nonlinear problem since we include collision handling and updated kernel values in our iteration process. As a final step, the velocities and positions of the next physics update step are computed. The PCISPH method is illustrated in Algorithm 2.

3.2.2 Pressure Derivation

One of the main difficulties is to derive the pressure change from the predicted density variation (line 18 of Algorithm 2). This pressure update is executed in each iteration, reducing the density fluctuation of the particle. The aim is to find

Algorithm 2 PCISPH

```
1  while animating do
2    for all i do
3      find neighborhoods N_i(t)
4    end for
5    for all i do
6      compute forces F^{v,g,ext}(t)
7      initialize pressure p(t) = 0.0
8      initialize pressure force F^p(t) = 0.0
9    end for
10   while (ρ*_err(t+1) > η) || (iter < minIterations) do
11     for all i do
12       predict velocity v*_i(t+1)
13       predict position x*_i(t+1)
14     end for
15     for all i do
16       predict density ρ*_i(t+1)
17       predict density variation ρ*_err(t+1)
18       update pressure p_i(t)+= f(ρ*_err(t+1))
19     end for
20     for all i do
21       compute pressure force F^p(t)
22     end for
23   end while
24   for all i do
25     compute new velocity v_i(t+1)
26     compute new position x_i(t+1)
27   end for
28 end while
```

a pressure p which changes the particle positions in such a way that the predicted density corresponds to the reference density. Over the course of this section, a set of approximations will be made to derive a simple update rule for the pressure (Equations 3.7 to 3.9). Although the approximations increase the number of convergence iterations which are needed until the desired density fluctuation limit is reached, they keep the final pressure update rule simple and thus efficient to compute.

For a given kernel smoothing length h, the density at a point in time $t + 1$ is computed using the SPH density summation equation analogously to Equa-

tion 2.13

$$\begin{aligned}\rho_i(t+1) &= m\sum_j W(\mathbf{x}_i(t+1)-\mathbf{x}_j(t+1))\\ &= m\sum_j W(\mathbf{x}_i(t)+\Delta\mathbf{x}_i(t)-\mathbf{x}_j(t)-\Delta\mathbf{x}_j(t))\\ &= m\sum_j W(\mathbf{d}_{ij}(t)+\Delta\mathbf{d}_{ij}(t))\end{aligned}$$

where $\mathbf{d}_{ij}(t) = \mathbf{x}_i(t) - \mathbf{x}_j(t)$, and $\Delta\mathbf{d}_{ij}(t) = \Delta\mathbf{x}_i(t) - \Delta\mathbf{x}_j(t)$. Assuming that $\Delta\mathbf{d}_{ij}$ is relatively small, the first order Taylor approximation can be applied to the term $W(\mathbf{d}_{ij}(t) + \Delta\mathbf{d}_{ij}(t))$ resulting in

$$\begin{aligned}\rho_i(t+1) &= m\sum_j W(\mathbf{d}_{ij}(t)) + \nabla W(\mathbf{d}_{ij}(t))\cdot\Delta\mathbf{d}_{ij}(t)\\ &= m\sum_j W(\mathbf{x}_i(t)-\mathbf{x}_j(t)) +\\ &\quad m\sum_j \nabla W(\mathbf{x}_i(t)-\mathbf{x}_j(t))\cdot(\Delta\mathbf{x}_i(t)-\Delta\mathbf{x}_j(t))\\ &= \rho_i(t)+\Delta\rho_i(t).\end{aligned}$$

In this equation, the term $\Delta\rho_i(t)$ is unknown and, as we show later, a function of p which we are looking for. After reformulation and using $W_{ij} = W(\mathbf{x}_i(t) - \mathbf{x}_j(t))$ we get

$$\begin{aligned}\Delta\rho_i(t) &= m\sum_j \nabla W_{ij}\cdot(\Delta\mathbf{x}_i(t)-\Delta\mathbf{x}_j(t))\\ &= m\Big(\sum_j \nabla W_{ij}\Delta\mathbf{x}_i(t) - \sum_j \nabla W_{ij}\Delta\mathbf{x}_j(t)\Big)\\ &= m\Big(\Delta\mathbf{x}_i(t)\sum_j \nabla W_{ij} - \sum_j \nabla W_{ij}\Delta\mathbf{x}_j(t)\Big)\end{aligned} \tag{3.1}$$

$\Delta\mathbf{x}$ can be derived from the time integration scheme (Leap-Frog). Neglecting all forces but the pressure force we get

$$\Delta\mathbf{x}_i = \Delta t^2\frac{\mathbf{F}_i^p}{m}. \tag{3.2}$$

If we make the simplistic assumption that neighbors have equal pressures $\tilde{p}_i$ and that the density corresponds to the rest density ρ_0 (according to the incompressibility condition), this results in

$$\mathbf{F}_i^p = -m^2\sum_j(\frac{\tilde{p}_i}{\rho_0^2}+\frac{\tilde{p}_i}{\rho_0^2})\nabla W_{ij} = -m^2\frac{2\tilde{p}_i}{\rho_0^2}\sum_j \nabla W_{ij}. \tag{3.3}$$

Inserting Equation 3.3 into Equation 3.2 we get

$$\Delta \mathbf{x}_i = -\Delta t^2 m \frac{2\tilde{p}_i}{\rho_0^2} \sum_j \nabla W_{ij}. \tag{3.4}$$

Due to the pressure p_i of particle i the position of a neighboring particle changes by $\Delta \mathbf{x}_{j|i}$. As the pressure forces are symmetric, particle j gets the following contribution from i

$$\mathbf{F}^p_{j|i} = m^2(\frac{\tilde{p}_i}{\rho_0^2} + \frac{\tilde{p}_i}{\rho_0^2})\nabla W_{ij} = m^2 \frac{2\tilde{p}_i}{\rho_0^2} \nabla W_{ij},$$

and the position of j changes by

$$\Delta \mathbf{x}_{j|i} = \Delta t^2 m \frac{2\tilde{p}_i}{\rho_0^2} \nabla W_{ij}. \tag{3.5}$$

Note that we only consider the effect of the central particle i here, i.e. $\Delta \mathbf{x}_j = \Delta \mathbf{x}_{j|i}$. Equation 3.4 and Equation 3.5 can now be inserted into Equation 3.1 resulting in

$$\begin{aligned}
\Delta \rho_i(t) &= m\Big(-\Delta t^2 m \frac{2\tilde{p}_i}{\rho_0^2} \sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} - \\
&\quad \sum_j (\nabla W_{ij} \cdot \Delta t^2 m \frac{2\tilde{p}_i}{\rho_0^2} \nabla W_{ij})\Big) \\
&= \Delta t^2 m^2 \frac{2\tilde{p}_i}{\rho_0^2}\Big(-\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} - \\
&\quad \sum_j (\nabla W_{ij} \cdot \nabla W_{ij})\Big)
\end{aligned}$$

After solving for $\tilde{p}_i$ we get

$$\tilde{p}_i = \frac{\Delta \rho_i(t)}{\beta(-\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} - \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}))} \tag{3.6}$$

where β is

$$\beta = \Delta t^2 m^2 \frac{2}{\rho_0^2}.$$

The meaning of Equation 3.6 is that a pressure $\tilde{p}_i$ is needed to achieve a change in density of $\Delta \rho_i(t)$. As we know the predicted density error $\rho^*_{err_i} = \rho^*_i - \rho_0$ of a particle, we can thus reverse that error by applying a pressure of

$$\tilde{p}_i = \frac{-\rho^*_{err_i}}{\beta(-\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} - \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}))}.$$

This formula shows problems in situations where i is suffering from particle deficiency in the neighborhood, which can be observed for example at domain boundaries and at the free surface of the fluid [Chen et al., 1999]. In these situations, the SPH equations result in falsified values. To circumvent that problem, we precompute a single scaling factor δ according to the following formula which is evaluated for a prototype particle with a filled neighborhood. The resulting value is then used for all particles. Finally, we end up with the following equations which are used in the PCISPH method

$$\delta = \frac{-1}{\beta(-\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} - \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}))} \tag{3.7}$$

and

$$\tilde{p}_i = \delta \rho^*_{err_i}. \tag{3.8}$$

Since we repeat the prediction-correction step as long as the incompressibility condition is not yet satisfied, the correction pressures of the individual iterations are accumulated as indicated on line 18 of Algorithm 2

$$p_i \mathrel{+}= \tilde{p}_i. \tag{3.9}$$

3.2.3 Implementation

Neighborhood Approximation

Before predicting the density $\rho^*_i(t+1)$ of a particle (line 16 of Algorithm 2), the neighborhood should be recomputed using the predicted positions $\mathbf{x}^*(t+1)$. However, for efficiency reasons we reuse the current neighbors $N_i(t)$ at time t and only recompute the distances and the kernel values. This approximation leads to small errors in the density and pressure estimates. In the case of density overestimation the final real densities show lower fluctuations than the requested threshold η. In the opposite case – density underestimation – the correction loop might be aborted prematurely. Such situations are not yet handled in the current implementation but can be avoided by using sufficiently small time steps, or by recomputing the neighborhoods in these particular situations.

Information Propagation

To limit temporal fluctuations in the resulting pressure field we found it advantageous to employ a minimum number of iterations in the pressure update loop. This gives the particles enough time to propagate information about predicted particle locations. We found a minimum of 3 iterations generally sufficient to achieve a low level of pressure fluctuations.

3.3 Results

3.3.1 Performance Comparison

We set up a test scene (Figure 3.4) to compare the simulation times and visual results of both the commonly used WCSPH and our new PCISPH method. The performance measurements and simulation data are summarized in Table 3.1. All timings are given for an Intel Core2 2.66 GHz CPU.

We executed different simulation runs with varying particle resolutions (10K and 100K) and varying error threshold η (1% and 0.1%) which defines the maximally allowed density fluctuation from the reference density. The 10K and 100K examples have corresponding scene setups but different fluid discretizations, meaning that a particle in the 10K example represents a larger fluid volume than one in the 100K example. Since in SPH a particle always needs to have around 30-40 neighbors, the support radius has to be increased with increasing particle volume, which in turn influences the time step size. The time step is set according to a CFL condition where the force terms, the stiffness parameter k, and the viscous term are involved [Monaghan, 1992]. While in WCSPH the time step is dominated by k, it has no influence in PCISPH and can be omitted. Thus, for low viscosity fluids, the time step in PCISPH is dominated by the force terms, allowing significantly larger time steps than those used in WCSPH. The stiffness parameter k of WCSPH was set in such a way that η was satisfied, which was found to be $k = 7 \cdot 10^4$ for $\eta = 1\%$, and $k = 6 \cdot 10^6$ for $\eta = 0.1\%$. In contrast, PCISPH does not have to cope with finding an appropriate stiffness value since the desired η can be specified directly.

In the case of η=1%, the time step of PCISPH is determined to be in fact 35 times larger than the one of WCSPH. With a smaller η the difference is even larger, $\eta = 0.1\%$ leads to an increase of the time step for PCISPH by a factor of 151. While in WCSPH the computation time per simulation step stays more or less constant, it varies in PCISPH since the time per simulation step depends on the number of executed convergence iterations. Therefore, we compare the overall computation time of WCSPH and PCISPH over the entire simulated time period. Although the cost per physics time step is higher with PCISPH than WCSPH, the overall speed-up over WCSPH still reaches a factor of 15 and 16 for $\eta = 1\%$ and 55 for $\eta = 0.1\%$, respectively (Figure 3.2).

3.3.2 Convergence Analysis

In the previously described test scenes, the average number of convergence iterations executed per physics step is between 3.24 and 4.46. Note that the particle resolution has no effect on the average number of iterations. For the simulation

Model	η [%]	#p	k	Δt [s]	Δt ratio [s]	$Iter_{avg}$	t_{sim} [min]	speed-up
WCSPH	1.0	10K	$7{\cdot}10^4$	3.78e-5	-	- -	142.05	-
PCISPH	1.0	10K	-	0.0013	35	3.24	9.37	15.2
WCSPH	1.0	100K	$7{\cdot}10^4$	1.78e-5	-	-	4941.5	-
PCISPH	1.0	100K	-	0.00062	35	3.49	297.7	16.6
WCSPH	0.1	10K	$6{\cdot}10^6$	4.08e-6	-	-	1327.66	-
PCISPH	0.1	10K	-	0.00062	151.96	4.46	23.97	55.39

Table 3.1: *Comparison of PCISPH and WCSPH. The stiffness value k of WCSPH is chosen so that the density fluctuations are below η, and the time step size is determined according to the CFL condition. With our PCISPH method, a speed-up of a factor of 15 and 16 over WCSPH is reached with a maximal density fluctuation of η=1%. By restricting the error to η=0.1%, PCISPH reduces the computation time by a factor of 55.*

run with 100K particles the average number of iterations is plotted over time in Figure 3.3(a). The end time of 8s corresponds to the simulated real time.

The peaks indicate particle collisions with the ground and the side walls as in such situations larger density errors are predicted. Figure 3.3(b) shows several examples of the convergence within a single physics update step. It can be seen that the density error is approximately halved after the first iteration and continuously reduced in the following iterations until the error drops below η. In our experience this algorithm proved to be very robust and we did not encounter any divergence problems. However, it is likely that certain particle configurations exist that might show such problems.

3.3.3 Visual Result

The physical and visual results of WCSPH and PCISPH are compared in Figure 3.4. In this scene, the fluid is represented by 100K particles and a maximal density fluctuation of 1% is enforced. The comparison shows nicely that the PCISPH computations are in full agreement with the WCSPH results with only very minor detail differences. In this particular example, we reached a speed-up over WCSPH of a factor of 15.

When limiting the overall simulation time to 298min we have to reduce the particle number with WCSPH to 17k particles so that the simulation finishes within the given time constraint. With PCISPH, a resolution of 100K particles can be simulated within the given time (see the corresponding entry in Table 5.1). The comparison of the simulation results is shown in Figure 3.5. It can be seen that the lower particle resolution leads to less surface details and notably damped fluid movement.

A higher resolution example computed with PCISPH is shown in Figure 3.6

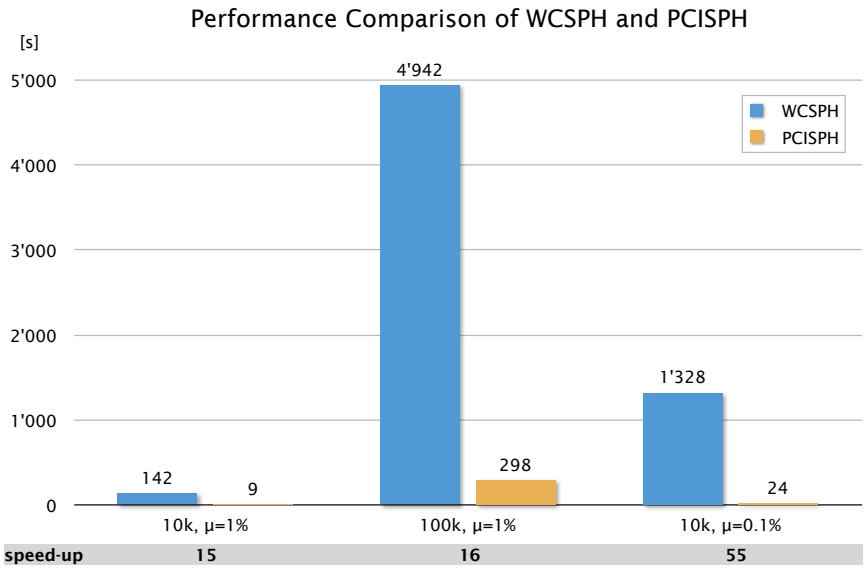


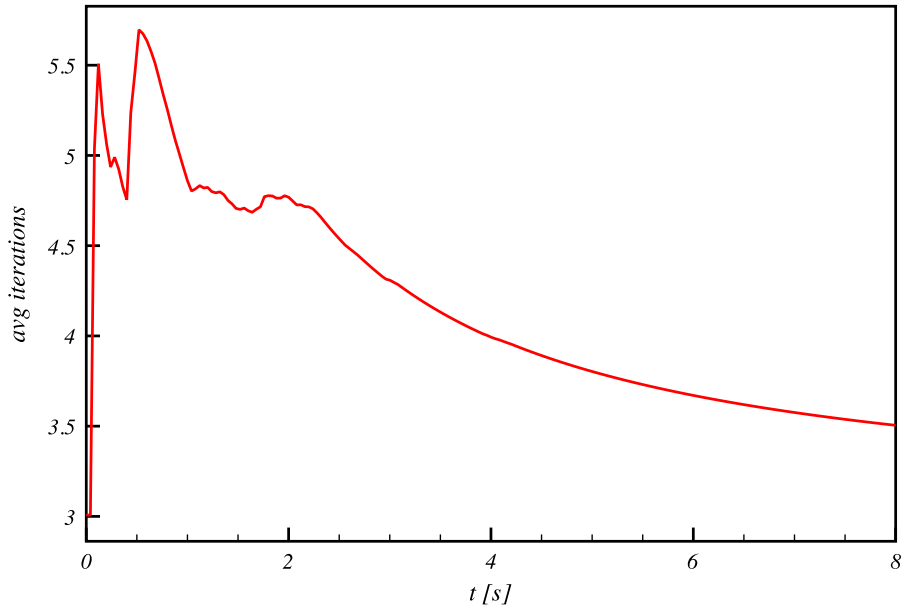
Figure 3.2: *Performance comparison of WCSPH and PCISPH according to Table 3.1. With PCISPH, a speed-up over WCSPH of a factor of 15 and 16 for $\eta = 1\%$ and 55 for $\eta = 0.1\%$, respectively, can be achieved.*

and Figure 3.7 where a wave generator agitates a water body consisting of 700K particles to interact with cylindrical obstacles in a tank. Collisions with the obstacles and the domain boundaries represent situations where large density variations are predicted and more iterations have to be executed. In Figure 3.8, 2M particles are used to simulate the collapsing column example with PCISPH. In both of these simulations, a η of 1% is enforced which eliminates compression artifacts and enables realistic wave breaking and splashing behavior. In all examples, the surface of the fluid is reconstructed and rendered with the raytracing approach presented in Section 6.3.

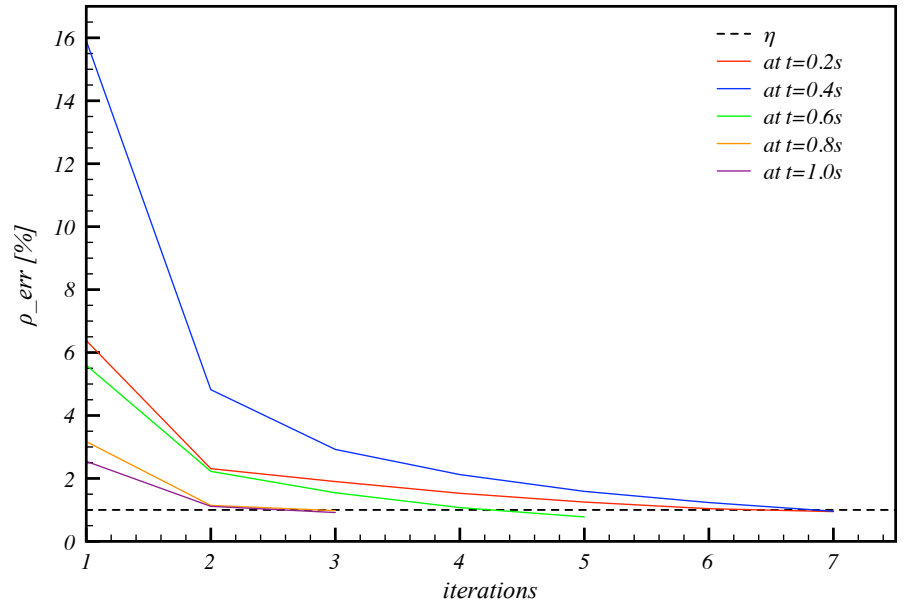
3.4 Discussion

One issue of the current implementation is the neighborhood approximation which can lead to underestimated density errors aborting the convergence loop prematurely as we have discussed in Section 3.2.3. This problem can be addressed by detecting such situations and adapting the time step size or recomputing the

neighbors in this particular simulation step. Besides that, our current implementation does not yet account for the particle deficiency near boundaries. In these situations, density values are falsified and compression artifacts can occur. The inclusion of ghost particles in the density computation or the use of the *Corrective Smoothed Particle Method* presented in [Chen et al., 1999] can solve this problem.



(a) Average number of convergence iterations over time. After 8s of simulated real time, an average of 3.49 is reached.



(b) Several convergence examples at different points in time t.

Figure 3.3: *Convergence statistics of the 100K particles simulation shown in Table 3.1 and Figure 3.4.*

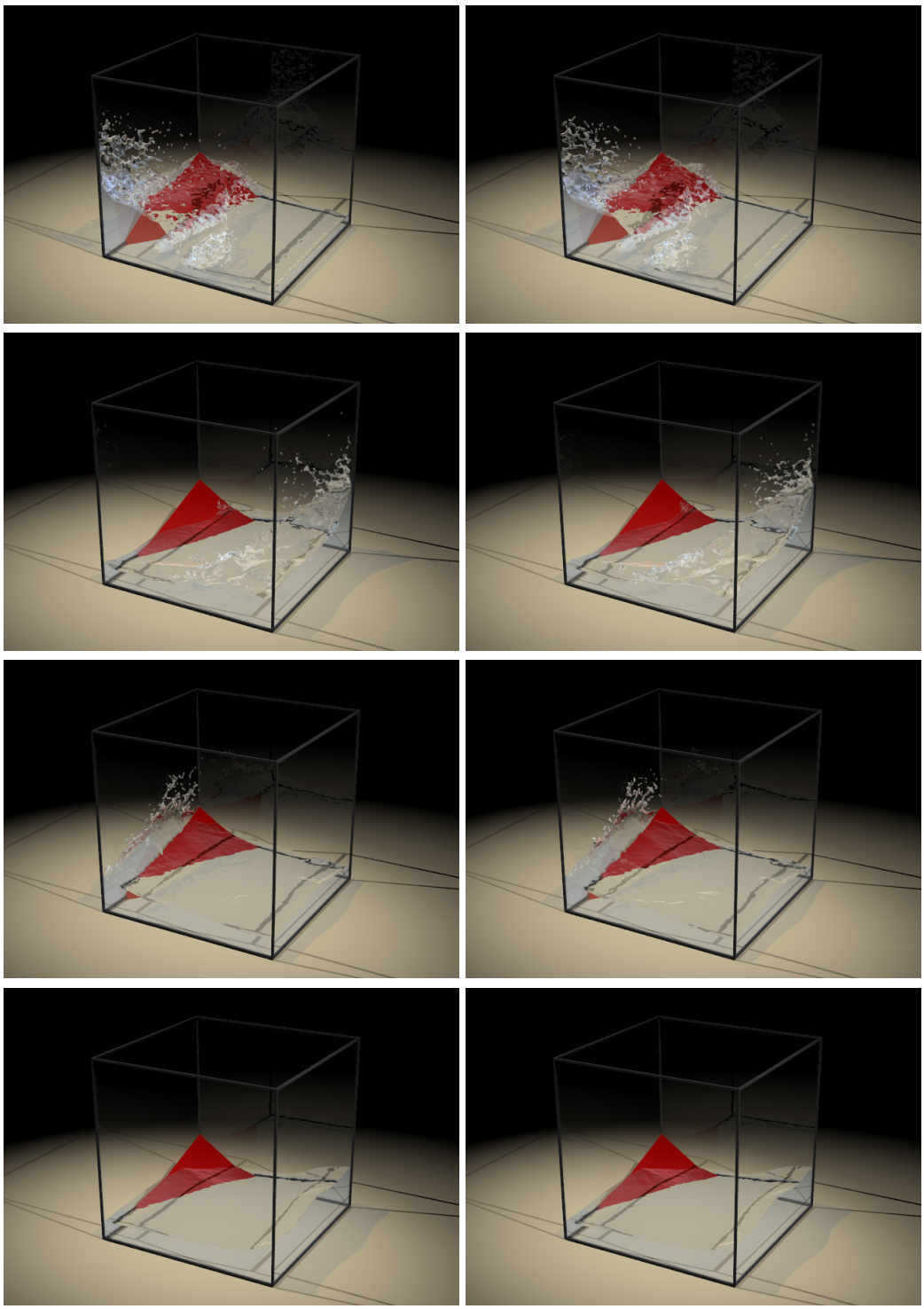


Figure 3.4: *Side-by-side comparison of a fluid discretized by 100K particles at time $t = 1.3s$, $t = 2.4s$, $t = 3.2s$, and $t = 4.5s$. The example is simulated with WCSPH (left) and PCISPH (right), respectively. The computations correspond to the statistics given in Table 5.1 for the 100K particles simulation.*

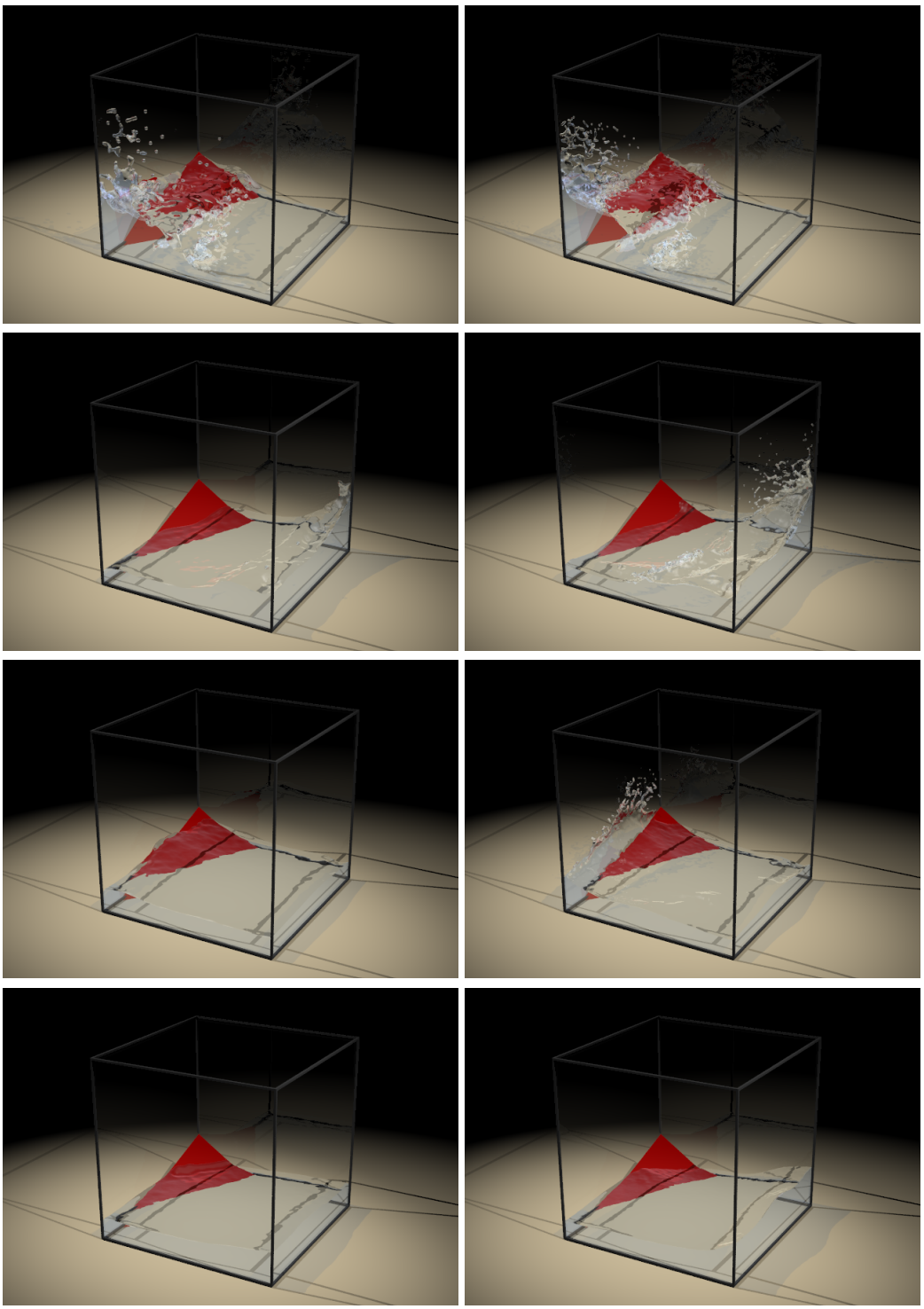


Figure 3.5: *Comparison of WCSPH (left, 17k particles) and PCISPH (right, 100K particles) with equal computation times. The rows correspond to the times* $t = 1.3s$, $t = 2.4s$, $t = 3.2s$, *and* $t = 4.5s$.

Figure 3.6: *Wave breaking and splashing in a wave tank simulated with the proposed incompressible PCISPH method.*

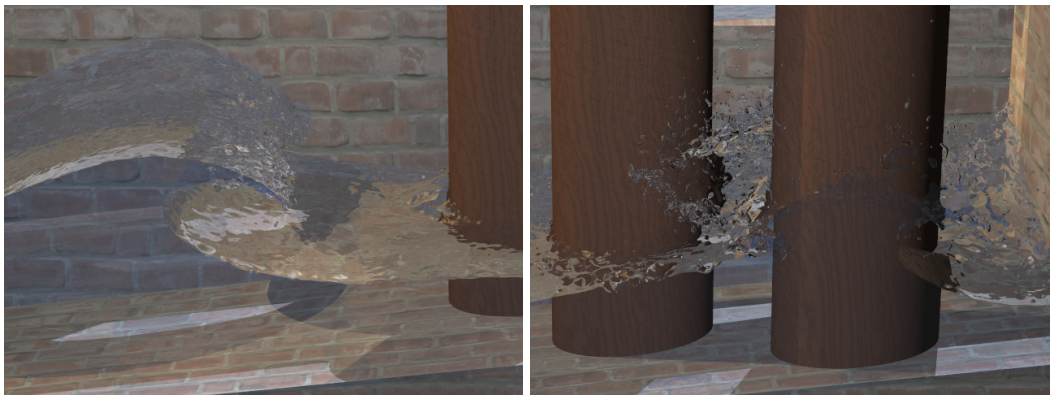


Figure 3.7: *Close-up views of the wave tank containing 700K particles simulated with our incompressible PCISPH method.*

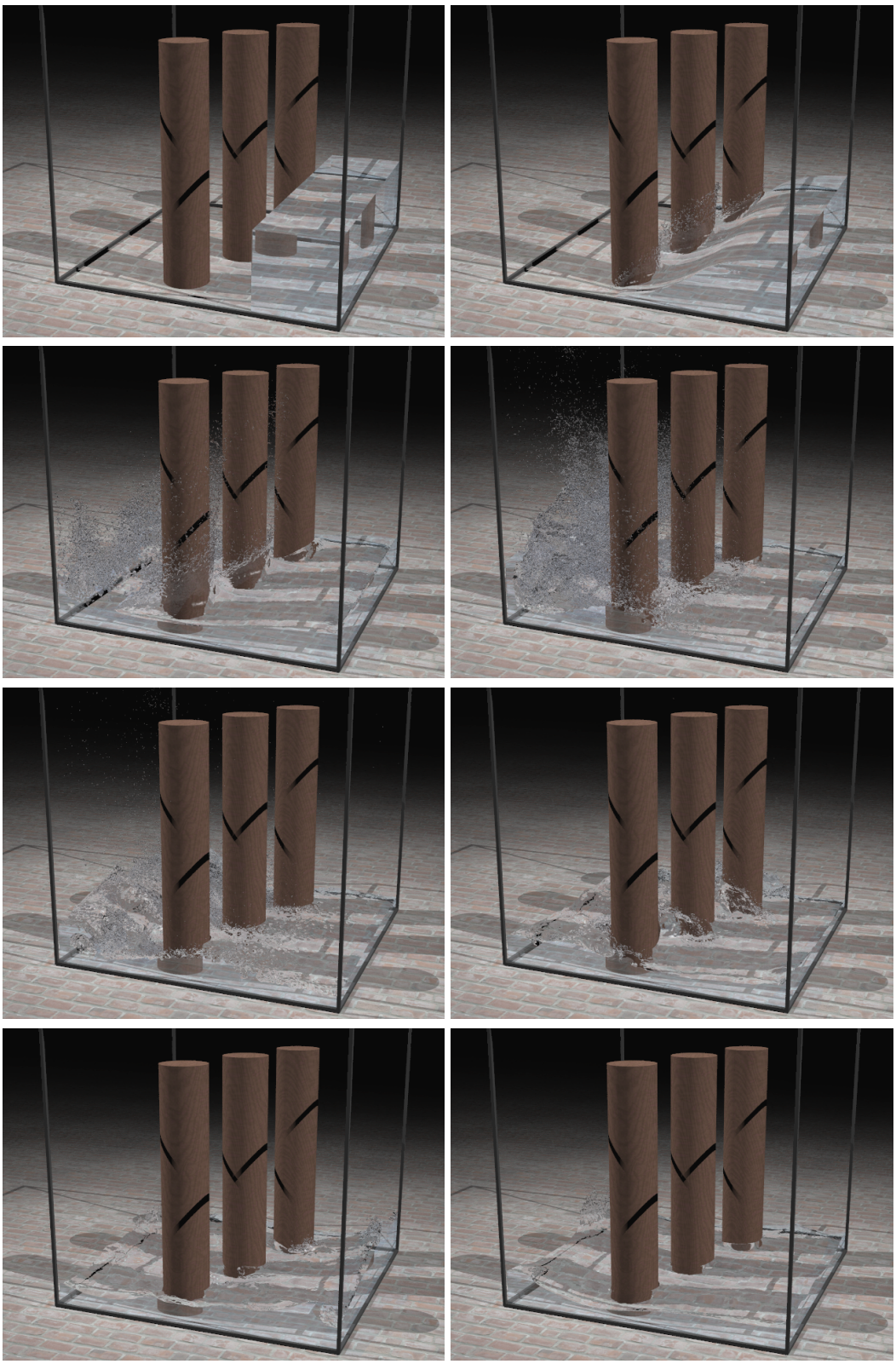


Figure 3.8: *PCISPH simulation of 2M particles interacting with cylinder obstacles.*

CHAPTER 4

INTERFACE TENSION

4.1 Multiple Interacting Fluids

When simulating fluids, it is important to capture interaction effects accurately in order to reproduce real world behavior. Focusing on the interaction between multiple fluids, the challenges are to realistically model miscible as well as immiscible liquids. In that context, we can observe that surface tension forces produce effects observable in everyday life. Some examples are the formation of drops, puddles on a surface, soap bubbles, and separation of dissimilar liquids such as oil and water.

So far, multiple fluids have been modeled using Eulerian as well as Lagangrian simulations. Although the strength of grid-based methods are the smooth and visually appealing surfaces, difficulties still exist in resolving small-scale features on or below the scale of the underlying grid. It is also clear that these methods still demand more attention to avoid the severe volume loss encountered, especially when simulating several turbulent liquids [Losasso et al., 2006b; Losasso et al., 2008]. Another approach is to use a fully particle-based fluid model such as SPH (Smoothed Particle Hydrodynamics) where particles with different physical quantities are used to represent several fluids [Müller et al., 2005b]. In contrast to level set methods, particle simulations need some effort to achieve smooth surfaces from the particles, but small-scale features down to single droplets are modeled implicitly, facilitating and enriching the simulation of complex interactions between multiple liquids.

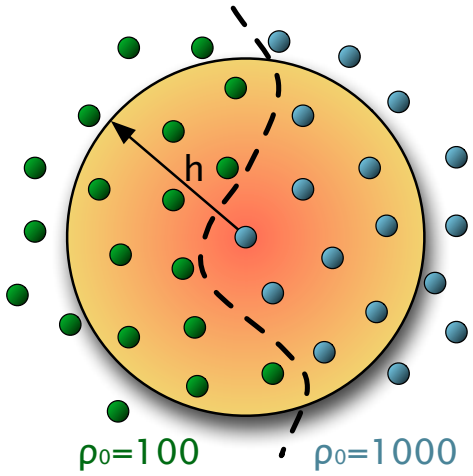


Figure 4.1: *Problems at interfaces of multiple fluids arise when rest densities and masses of neighboring particles vary within the smoothing length h since in such cases the resulting smoothed quantities show falsified values.*

In SPH, particles have a spatial distance (smoothing length) over which their properties are smoothed by a kernel function. Problems arise when rest densities and masses of neighboring particles vary within the smoothing length, as in such cases the smoothed quantities of a particle show falsified values. Such problems can be observed near the interface of multiple fluids with density contrasts. The erroneous quantities lead to undesirable effects, reaching from unphysical density and pressure variations to spurious and unnatural interface tensions (see for example the left image in Figure 4.2), and even to severe numerical instabilities.

In literature, these problems have been mainly described in computational physics so far, nevertheless, graphics applications have to cope with similar difficulties. In [Hoover, 1998], the spurious interface tension due to degraded densities and pressures near interfaces has been described for the first time. A similar observation was reported in [Agertz et al., 2006], where it has been shown that the erroneous pressure forces lead to a gap between two fluids with high density contrasts preventing important instabilities such as Kelvin-Helmholtz to evolve. When increasing the density contrast of the fluids, one has additionally to cope with severe numerical instabilities. [Colagrossi and Landrini, 2002] have shown that density ratios of more than a factor of 10 between two fluids cannot be stably simulated with SPH and that decreasing the time step does not reduce or overcome this problem. Alternatively to the standard density summation, the authors evolve the density over time according to the SPH equation for continuity (convergence

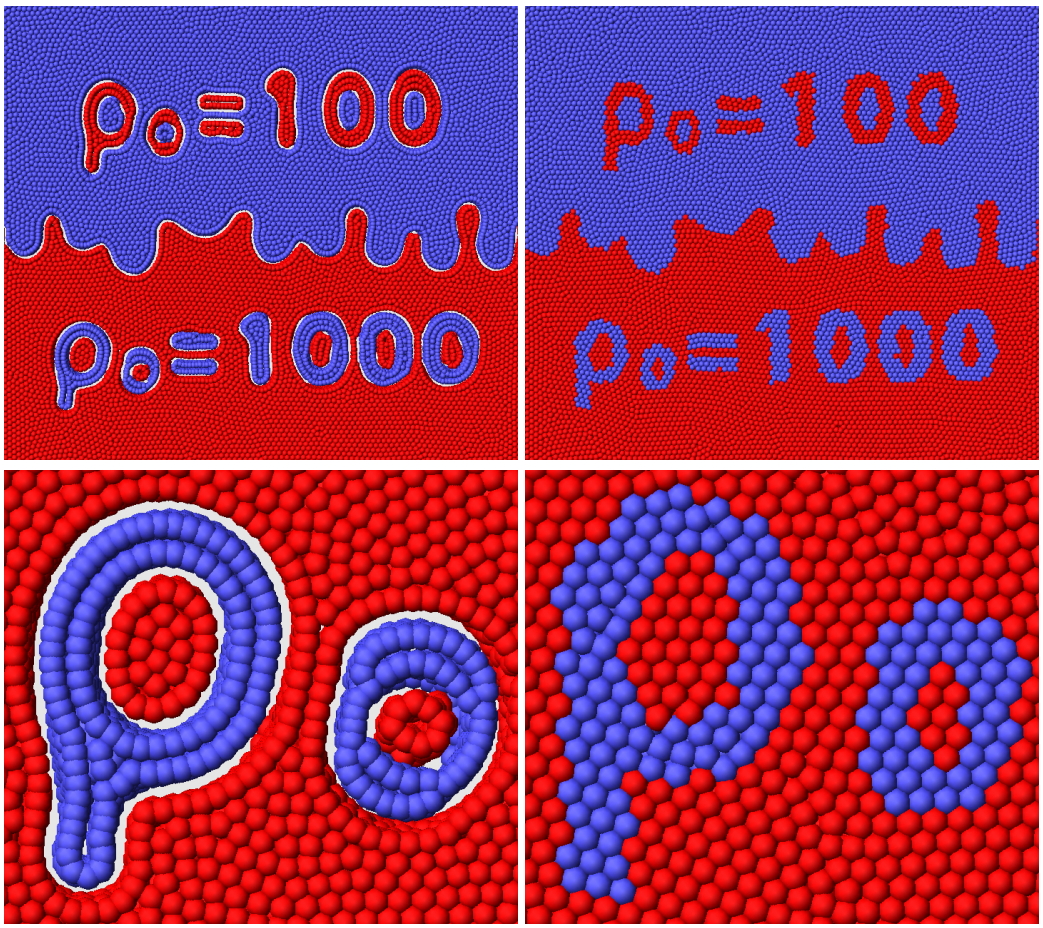


Figure 4.2: *In contrast to the new method (right), the use of standard SPH produces a spurious interface tension and a gap between two fluids with a density contrast (left).*

equation) [Monaghan, 1994]. Although the use of the convergence equation allows to set the initial densities freely, similar problems are encountered as when calculating the density directly from the particle distribution [Ott and Schnetter, 2003].

Because of the different requirements of computational physics and computer graphics, this thesis focuses on 3D simulations and visually demonstrates how the unnatural interface tension of the standard SPH formulation behaves. Our examples highlight the fact that when using standard SPH a user has no control over the behavior of multiple fluids and that in the worst case the simulation results in instability. As one of the main issues in graphics is to have full control over the simulated materials, we introduce a method which can handle interface disconti-

nuities and eliminates the artifacts described above. Since our derived equations are simple modifications of a standard SPH solver, they are easy to implement and do not negatively affect the performance. In the following, we propose to compute the density based on the particle number density and we derive new formulations for the pressure equation, pressure forces as well as viscous forces. Additionally, a new interface tension model based on a smoothed and normalized color field is introduced, adding a fully controllable interface tension to our model. This allows us to simulate miscible as well as immiscible fluids according to the simulation problem of interest.

Similar to us, [Ott and Schnetter, 2003; Tartakovsky and Meakin, 2005; Hu and Adams, 2006] handle density discontinuities at interfaces of multiple fluids. [Ott and Schnetter, 2003] have derived an adapted continuity equation and they have compared sound and shock wave simulation results to analytical solutions. Although the results for these specific applications are promising, our experiments have shown that the use of the standard as well as the adapted continuity equation does not produce stable results for long-term simulations. This is due to severe density integration errors, especially when using large time steps and low-order time integration schemes which is important for the targeted type of applications. Both [Tartakovsky and Meakin, 2005] and [Hu and Adams, 2006] use a corrected density summation for their investigations. The former work concentrates on miscible flow in fracture apertures with complex geometry and combines a modified SPH flow equation with an advection-diffusion equation. Tension forces are not included in their model, and the pressure computation does only allow the simulation of closed systems or systems with periodic boundary conditions. The latter work focuses on the investigation of numerical examples such as droplet oscillation and deformation in shear flow in 2D and the comparison to analytical solutions. This work has been extended with an incompressibility condition in [Hu and Adams, 2007].

Besides the works already mentioned above, earlier research on multi-phase fluid simulation methods includes [Kang et al., 2000; Pelupessy et al., 2003; Hong and Kim, 2005], addressing discontinuous properties, and [Hong and Kim, 2003; Greenwood and House, 2004; Mihalef et al., 2006; Zheng et al., 2006], focusing on bubbles and foam. While these techniques are all fully Eulerian, [Losasso et al., 2008] introduced a level set method which is coupled with SPH particles representing diffuse regions such as spray. A shallow water simulation using SPH particles to represent foam has been presented in [Thürey et al., 2007], and a pure particle simulation based on SPH to deal with multiple liquids and boiling effects has been demonstrated in [Müller et al., 2005b]. In the latter work, density ratios are kept small, reducing the visibility of the problems coming with multiple fluids. Immiscible fluids have been animated in [Mao and Yang, 2006] by explicitly detecting colliding particles.

4.2 Adapted SPH Equations for Miscible Fluids

4.2.1 Problem of Standard SPH

The standard SPH density summation (Equation 2.13)

$$\rho_i = \sum_j m_j W(\mathbf{x}_{ij}, h)$$

becomes problematic as soon as a particle has neighboring particles with different rest densities (and therefore different masses, as we require constant rest volumes throughout the particles). This is the case close to the interface of two fluids with a rest density contrast. For particles close to the interface, the computed density is underestimated if they belong to the fluid with higher rest density, and overestimated otherwise. This happens because the standard SPH formulation smoothes the density and cannot accurately represent sharp density changes as it would be desired. This is illustrated in Figure 4.3 (a) and (b) and visualized in the left part of Figure 4.4. The falsified densities induce wrong pressure values close to the interface (Figure 4.3 (c)), leading to a spurious interface tension and a large gap between the fluids (Figure 4.2). Even worse, the erroneous pressure forces induce numerical instabilities at the interface and make it impossible to simulate multiple fluids with high density ratios.

4.2.2 Comparison of Pressure Force Equations

Regarding the standard SPH formulation for multiple different fluids, not only the density is problematic but also the computation of the pressure forces. We compare the two techniques mainly used in graphics, which are (Equations 2.17 and 2.18)

$$\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i} \sum_j \frac{m_j}{\rho_j} \frac{p_i + p_j}{2} \nabla W(\mathbf{x}_{ij}, h)$$

and

$$\mathbf{F}_i^{pressure} = -\sum_j m_i m_j (\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}) \nabla W(\mathbf{x}_{ij}, h),$$

and derive adapted equations applicable to multiple fluids later on.

Our experiments have shown that for fluids with small density contrasts (density ratios of approximately a factor of 2), both pressure force equations result in almost the same behavior regarding spurious interface tension and the undesired gap between the fluids. When increasing the density contrast, the use of Equation 2.17 leads to unstable simulations which cannot be overcome even by decreasing the time step of the simulation significantly (Figure 4.5 (a) upper-left).

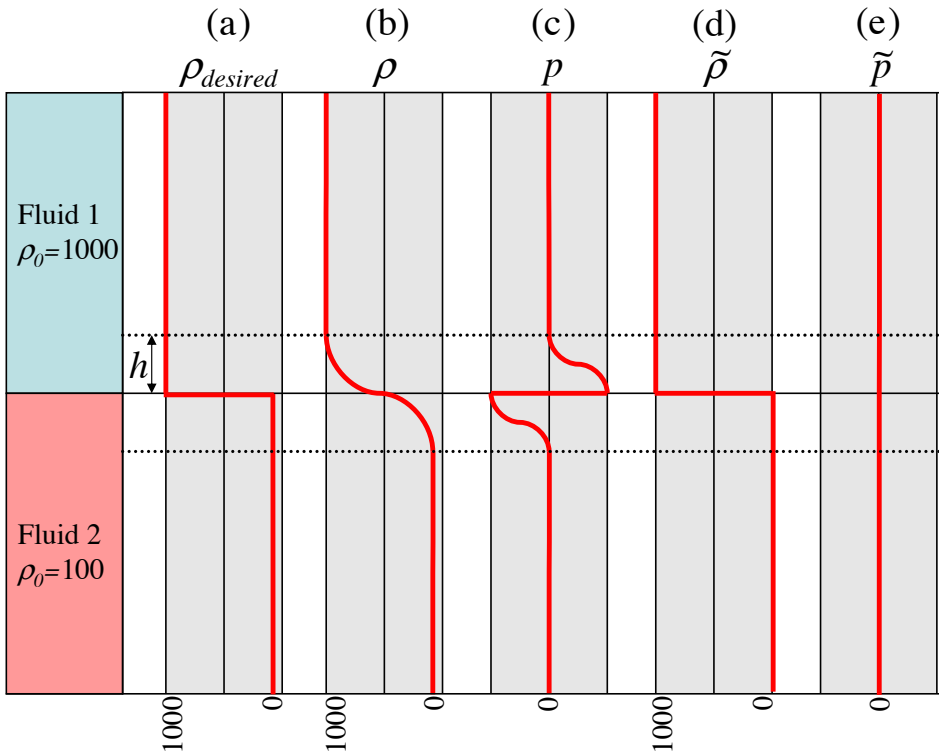


Figure 4.3: *Several physical quantities in a 1D example. Standard SPH cannot represent actual desired density discontinuities (a), as it smoothes the density over the interface (b). As a result, erroneous pressures are present near the interface (c). We derive new SPH equations using the particle density, resulting in densities (d) and pressures (e) with the desired behavior.*

For these tests, we used a viscosity coefficient μ of 5Ns/m^2 and a stiffness k of 1000Nm/kg. When using the parameters described in [Müller et al., 2005b] which are a μ of 20Ns/m^2 and a k of 20Nm/kg, the simulation of density ratios up to a factor of 10 is feasible, but it comes at the expense of undesired smoothing and compressibility effects. As we have shown in Chapter 3, a k of $7 \cdot 10^4$ Ns/m had to be used to keep density fluctuations in our simulation example below 1%.

The simulation behaves differently when using Equation 2.18 as it is stable up to a density ratio of 10 (Figure 4.5 (a) lower-left). Larger density contrasts are not stable, and reducing the time step has again no effect onto the stability.

In Section 4.2.4, we present the modified equations for both pressure force equations and we show that the numerical instabilities and spurious interface tensions are eliminated for both approaches (Figure 4.5 (b)).

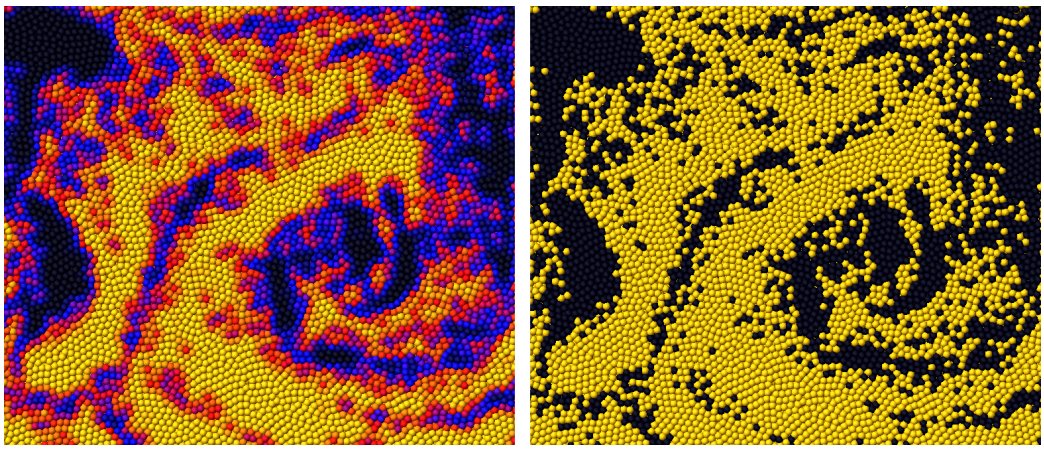


Figure 4.4: *Standard SPH density (left) versus our corrected density (right). The two fluids have rest densities of 1000kg/m^3 and 100kg/m^3. The computed density is color-coded with yellow being 1000, red 700, blue 400, and black 100, respectively.*

4.2.3 Density Model

To handle density discontinuities at interfaces between multiple fluids with varying rest densities correctly, we propose to replace the standard density summation given by Equation 2.13 by a measure of particle density (sometimes called number density), similar to [Ott and Schnetter, 2003; Premoze et al., 2003; Tartakovsky and Meakin, 2005; Hu and Adams, 2006]. The idea is to make each particle treat its neighbors as if they would have the same rest density and mass as itself. The particle density δ_i of a particle is defined as

$$\delta_i = \sum_j W(\mathbf{x}_{ij}, h). \tag{4.1}$$

We then compute the adapted physical fluid density $\tilde{\rho}_i$ of a particle by multiplying the particle density by the mass of the particle

$$\tilde{\rho}_i = m_i \delta_i = m_i \sum_j W(\mathbf{x}_{ij}, h). \tag{4.2}$$

For the volume V of a particle we then get

$$V_i = \frac{m_i}{\tilde{\rho}_i} = \frac{1}{\delta_i}. \tag{4.3}$$

For a single fluid where all particles have equal masses and rest densities, the presented formulation corresponds exactly to the standard SPH formulation. But

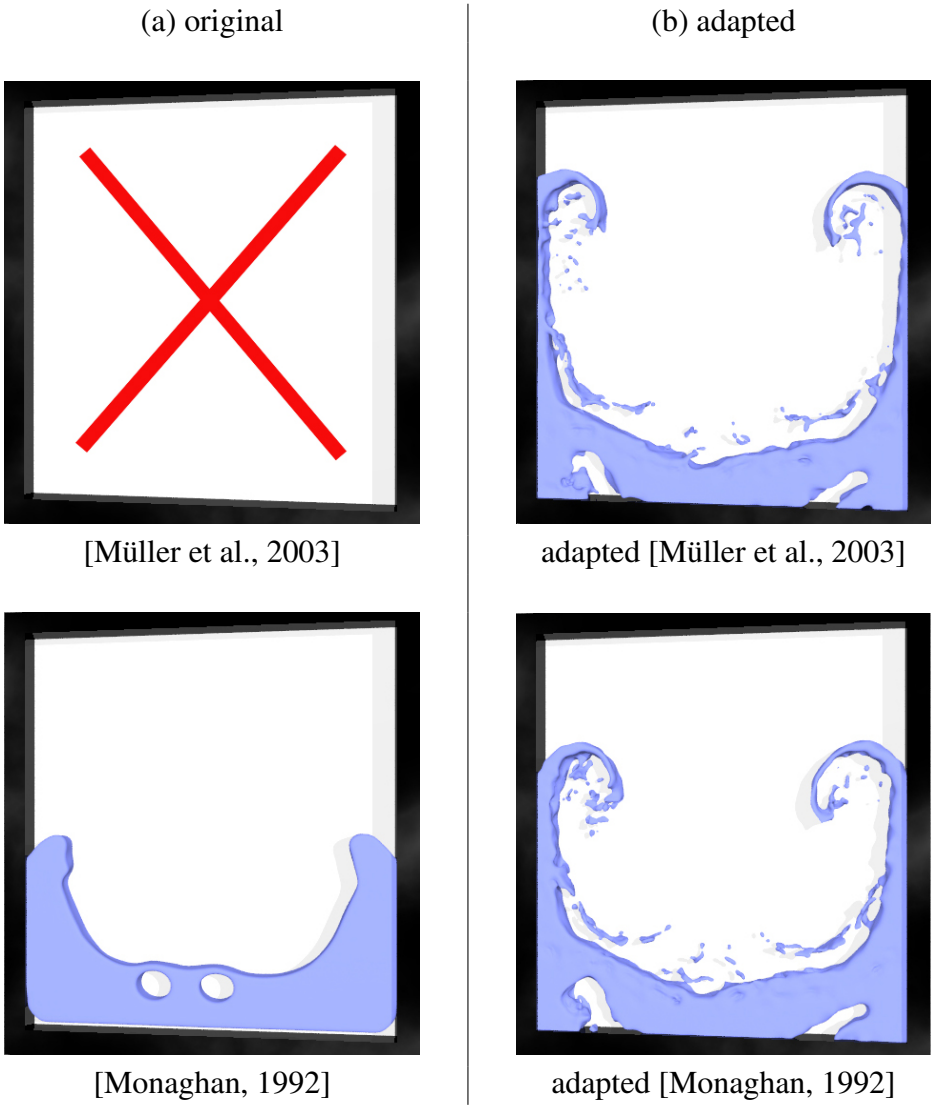


Figure 4.5: *Two fluids with a density ratio of 10 are simulated with two different pressure force equations, (a) on the left using the original formulations, and (b) on the right using our adapted equations. While the standard formulations result in instability (upper left: Equation 2.17) and spurious tension problems (lower left: Equation 2.18), these problems can be overcome by using our modified equations (Equation 4.7 and Equation 4.12).*

when dealing with multiple fluids of different densities we can achieve a density field reproducing sharp density changes at the interface of the fluids as shown in Figure 4.4 on the right.

4.2.4 Adapted Pressure and Pressure Forces

Following [Batchelor, 1967; Monaghan, 1994], we use the Tait equation to compute the pressure. In Equation 2.15, we replace the standard SPH density ρ by the adapted density $\tilde{\rho}$ introduced above, yielding the following equation for the pressure $\tilde{p}$

$$\tilde{p}_i = \frac{k\rho_0}{\gamma}((\frac{\tilde{\rho}_i}{\rho_0})^\gamma - 1). \tag{4.4}$$

Consequently, we can derive a new formulation for the pressure force. In the pressure gradient term $a = -\nabla p/\rho$ of the Navier Stokes equations we replace ρ by $\tilde{\rho}$ and p by $\tilde{p}$, yielding

$$a = -\frac{\nabla \tilde{p}}{\delta m}. \tag{4.5}$$

For the pressure force $\mathbf{F}^{pressure} = ma$ we then get

$$\mathbf{F}^{pressure} = -\frac{\nabla \tilde{p}}{\delta}. \tag{4.6}$$

When using the formulation of [Müller et al., 2003], the pressure force is derived by applying the SPH rules to ∇p and symmetrizing the equation. In the standard approach, this yields Equation 2.17. We derive the adapted pressure force equation similarly, but we again replace ρ by $\tilde{\rho}$ and p by $\tilde{p}$, yielding the final equation for the pressure force

$$\mathbf{F}_i^{pressure} = -\frac{1}{\delta_i}\sum_j \frac{1}{\delta_j}\frac{\tilde{p}_i + \tilde{p}_j}{2}\ \nabla W(\mathbf{x}_{ij}, h). \tag{4.7}$$

Monaghans pressure force equation [Monaghan, 1992] is derived differently from the one in [Müller et al., 2003]. In Monaghans derivation, the pressure gradient term of the Navier Stokes equations is symmetrized by applying the quotient rule

$$\frac{\nabla p}{\rho} = \nabla(\frac{p}{\rho}) + \frac{p}{\rho^2}\nabla\rho. \tag{4.8}$$

This formulation uses the density gradient $\nabla\rho$, which is problematic when applying our modified density $\tilde{\rho}$ for multiple fluids. In contrast to the standard SPH density ρ which was smoothed over the interface (Figure 4.3 (b)), $\tilde{\rho}$ is discontinuous (Figure 4.3 (d)) and the derivative thereof is thus not defined. Simply inserting $\tilde{\rho}$ and $\tilde{p}$ into Monaghans pressure force equation results in severe instabilities at the interface.

To solve this problem, we have to derive the pressure gradient in a different way. Our approach is to replace the discontinuous quantity $\nabla\rho$ by a C^1 continuous one. We use Equation 4.6 and apply the quotient rule. Thus, Equation 4.8 becomes

$$\frac{\nabla\tilde{p}}{\delta} = \nabla(\frac{\tilde{p}}{\delta}) + \frac{\tilde{p}}{\delta^2}\nabla\delta. \tag{4.9}$$

As can be seen, the discontinuous quantitiy $\nabla\rho$ is replaced by the continuous and derivable particle density. Applying the SPH rules, Equation 4.9 can be rewritten to

$$\frac{\nabla\tilde{p}}{\delta} = \sum_j (\frac{\tilde{p}_j}{\delta_j} + \frac{\tilde{p}_i}{{\delta_i}^2}\delta_j) V_j \nabla W(\mathbf{x}_{ij}, h). \tag{4.10}$$

In Equation 4.10, the volume V_j can be replaced by $1/\delta_j$, resulting in

$$\frac{\nabla\tilde{p}}{\delta} = \sum_j (\frac{\tilde{p}_j}{{\delta_j}^2} + \frac{\tilde{p}_i}{{\delta_i}^2}) \nabla W(\mathbf{x}_{ij}, h). \tag{4.11}$$

Finally, we get for the pressure force of a particle i

$$\mathbf{F}_i^{pressure} = -\sum_j (\frac{\tilde{p}_j}{{\delta_j}^2} + \frac{\tilde{p}_i}{{\delta_i}^2}) \nabla W(\mathbf{x}_{ij}, h). \tag{4.12}$$

In [Tartakovsky and Meakin, 2005; Hu and Adams, 2006], this formulation has been derived differently but adopted in a similar way.

4.2.5 Adapted Viscous Forces

We derive the adapted viscous forces by replacing the density ρ by the modified density $\tilde{\rho}$ in the viscosity term $\mu\nabla^2\mathbf{v}/\rho$ of the Navier-Stokes equations as well as in the derived SPH formulation. We end up with the following equation for the viscous force

$$\mathbf{F}_i^{viscosity} = \frac{1}{\delta_i} \sum_j \frac{\mu_i + \mu_j}{2} \frac{1}{\delta_j} (\mathbf{v}_j - \mathbf{v}_i)\ \nabla^2 W(\mathbf{x}_{ij}, h). \tag{4.13}$$

4.2.6 Controlling Interface Tension Forces

With the modified density, pressure, and force equations presented in the last sections we are able to eliminate all spurious and unnatural interface tension effects which are present when using the standard SPH method. Now we can introduce a fully controllable interface tension to our model, enabling a user to select the desired amount according to the simulation problem of interest.

Similar to [Morris, 2000], we use a color field to model tension forces. In contrast to their work and to [Müller et al., 2005b; Hu and Adams, 2006], we model the tension forces such that the free surface remains unaffected while the desired interface tension between any two different fluids can be controlled arbitrarily. If desired by the user, additional tension forces acting at the free surface can be simply added by using the technique presented in [Morris, 2000]. We define the interface tension force to be

$$\mathbf{F}_i^{interface} = \frac{1}{\delta_i}\theta\kappa\mathbf{n}, \tag{4.14}$$

where θ is the tension coefficient defining the strength of the force and $\mathbf{n}$ is the normal on the interface. This force acts to smooth interface regions of high curvature κ, in an attempt to minimize the total surface area. In order to compute $\mathbf{n}$ and κ, a color field is defined which is non-zero at all particle locations, and different color values are assigned to different fluid types. As suggested in [Morris, 2000], we smooth the color field to obtain more accurate estimates of the normals $\mathbf{n} = \nabla c$ afterwards. In order to avoid tension at the free surface, we additionally normalize the smoothed color value. Thus, the smoothed color value is given by

$$< c >_i = \frac{\sum_j \frac{1}{\delta_j} c_j W(\mathbf{x}_{ij}, h)}{\sum_j \frac{1}{\delta_j} W(\mathbf{x}_{ij}, h)}. \tag{4.15}$$

The accuracy of the normal can be improved additionally by using the difference between neighboring particle colors

$$\mathbf{n}_i = \sum_j \frac{1}{\delta_j}(< c >_j - < c >_i)\nabla W(\mathbf{x}_{ij}, h). \tag{4.16}$$

The curvature, which is defined as $\kappa = -\nabla \cdot \hat{\mathbf{n}}$, where $\hat{\mathbf{n}}$ is the unit normal, can be formulated with SPH and our adapted density as

$$\kappa = \frac{-\sum_j \frac{1}{\delta_j}(\hat{\mathbf{n}}_j - \hat{\mathbf{n}}_i) \cdot \nabla W(\mathbf{x}_{ij}, h)}{\sum_j \frac{1}{\delta_j} W(\mathbf{x}_{ij}, h)}. \tag{4.17}$$

4.3 Results

To demonstrate the effectiveness of our approach, we simulated several examples with varying resolution ranging from 20K to 1M particles on an Intel Core2 2.66 GHz. The computational cost for the examples range from 0.2s to 10s per time step and 20s to 40min to render one frame using the raytracing approach presented in Section 6.3. For all scenes, we used the leapfrog time integration scheme with

constant time step size throughout the simulation. The time step size was initially determined by using a CFL condition [Courant et al., 1967]. In our examples, this value was dominated by the stiffness of the fluid and was between 10e-3s and 10e-4s. Note that, compared to standard SPH, the time step size does not have to be decreased when using our method, and the cost per time step stays the same. Furthermore, our approach makes the simulation of high density ratios possible which cannot be stably simulated with standard SPH. An example where standard SPH failed in our tests is depicted in Figure 4.9, where 1 million particles representing 3 different fluids with a density ratio of 20 in total were simulated with our method. The margins are slightly cut to see the interior of the fluid.

Figures 4.6 and 4.7 depict another Rayleigh-Taylor instability with 80K particles representing two fluids with a density ratio of 10. Although we were able to simulate this example using Monaghans pressure equation, the result is suffering from severe and unnatural interface tension (Figure 4.6 (a), Figure 4.2). Our modifications eliminate all spurious interface tension effects (Figure 4.6 (b)), and allows us to explicitly add tension forces with full control over its strength (Figure 4.7 (c), (d)), facilitating the simulation of miscible and immiscible fluids. Figure 4.8 shows another example of how our interface tension between two fluids work.

In the last sections, we derived new equations for two different types of pressure force equations which are often used in graphics, allowing a user to select the desired formulation. Regardless of the type, the instability and spurious tension problems of the standard formulation (Figure 4.5, (a)) can be overcome by using our new method (Figure 4.5, (b)). While the standard SPH technique allows only the simulation of density ratios up to 2 or 10, respectively (depending on the type of pressure force equation as we have discussed in Section 4.2.2), our method enables the simulation of fluids with very high density ratios without having stability problems. This is demonstrated in Figure 4.10, where fluids with density ratios of up to 100 were simulated.

4.4 Discussion

Although our method overcomes the discontinuity problems at interfaces of multiple fluids, we would like to point out that other limitations of SPH remain. When dealing with large density ratios in SPH, the behavior of small, light volumes is negatively affected as the buoyancy is damped in specific situations. Although viscosity dampens turbulence and buoyancy to some extent, we have observed that this defect is apparent even without integrating any viscosity into the SPH model. We believe that this defect results from pressure forces compelling the particles to arrange in a stable equilibrium lattice structure [Lombardi et al., 1999]. As a

result, the buoyant volumes have to break open the crystallized particle configuration in order to rise. Thus the buoyancy may get weakened, most notably visible at small volumes and when the system comes to rest. Although this effect will need some attention in the future, this thesis focuses on the specific challenge of spurious and unphysical interface tension effects with the standard SPH approach only. Our proposed solution addresses this identified problem in such a way that no other aspects of SPH are seriously affected, being it its simulation performance or its advantages in modeling small-scale features and multiple materials, but also for that matter other disadvantages remain.

(a) (b)

Figure 4.6: *Two fluids with a density ratio of 10 at two different points in time. While standard SPH produces unnatural interface tension (a), our method prevents any spurious tension between the fluids (b).*

(c) (d)

Figure 4.7: *Two fluids with a density ratio of 10 at two different points in time. Since our method prevents any spurious tension between the fluids, interface tension forces can be added with full control, (c) and (d) show a tension strength of* $\theta = 5$ *and* $\theta = 35$*, respectively.*

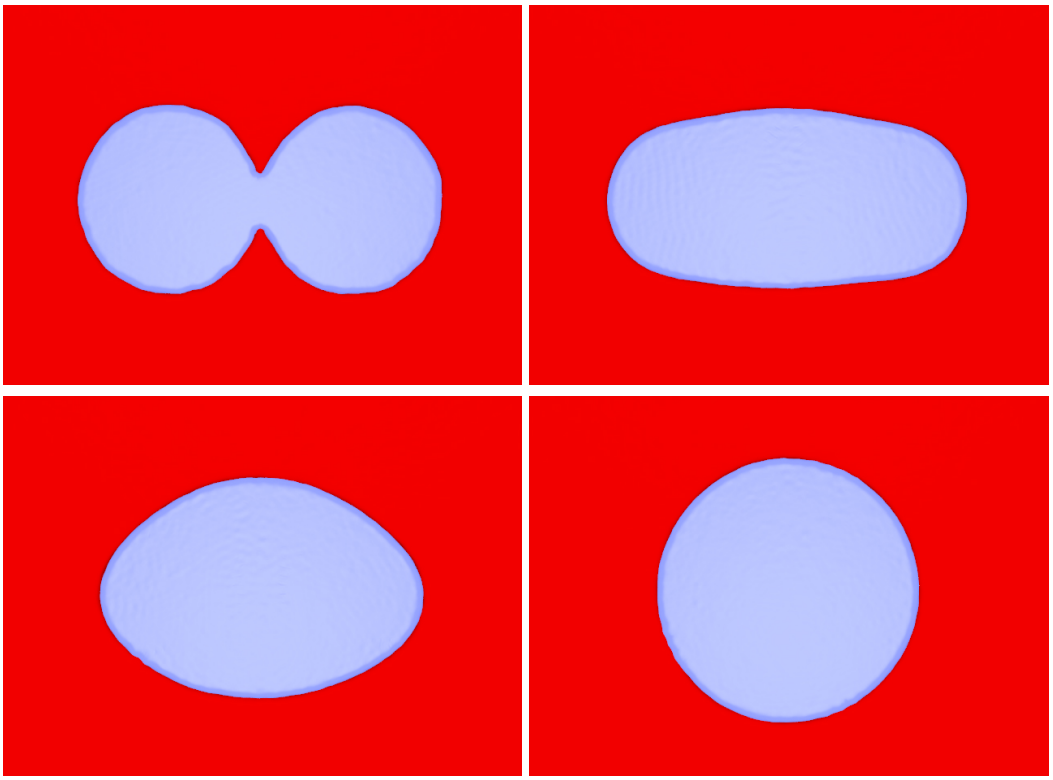


Figure 4.8: *The effect of interface tension acting between two fluids.*

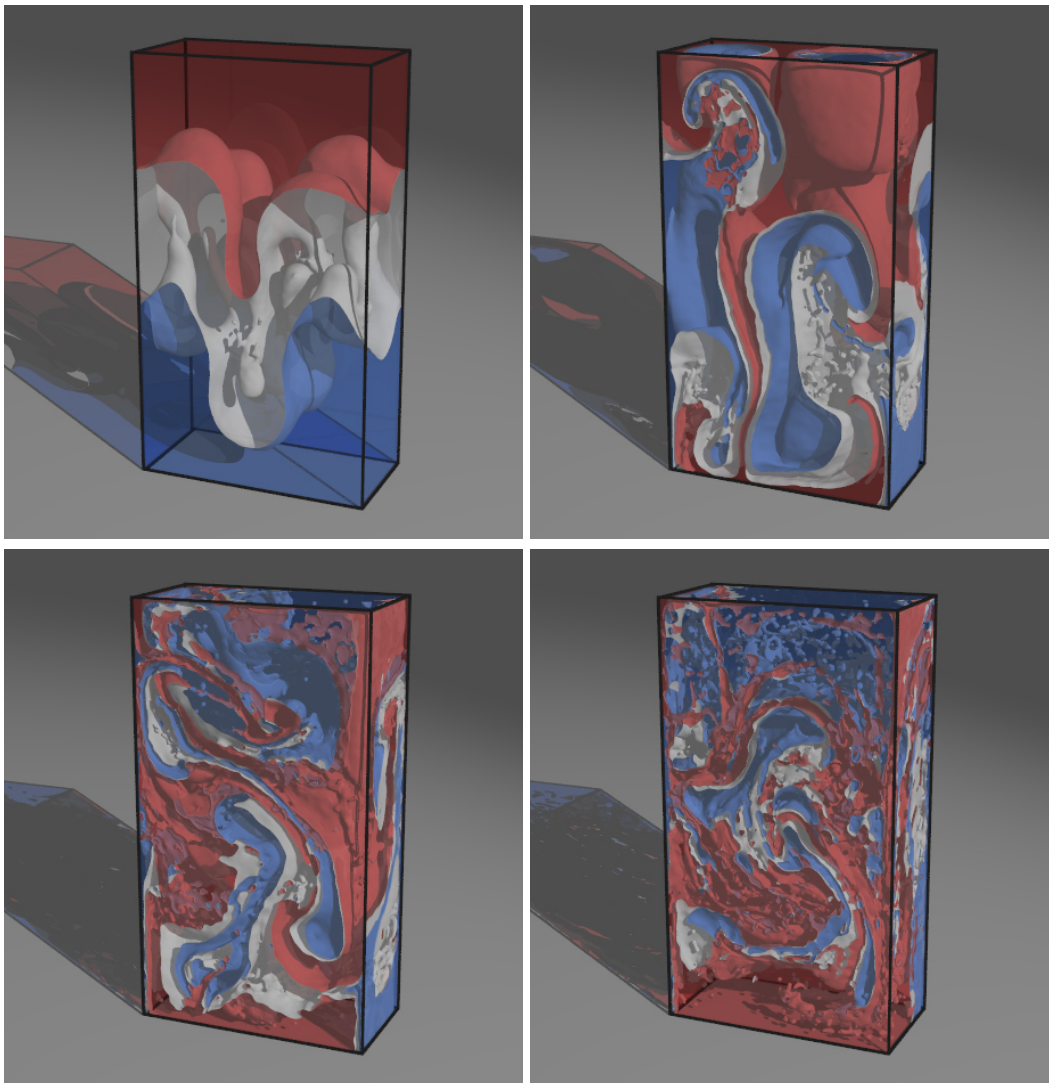


Figure 4.9: *Rayleigh-Taylor instability of three fluids with density contrasts simulated with our method using one million particles. The margins are slightly cut to see the interior of the fluid. In contrast to our method, standard SPH fails to stably simulate this example.*

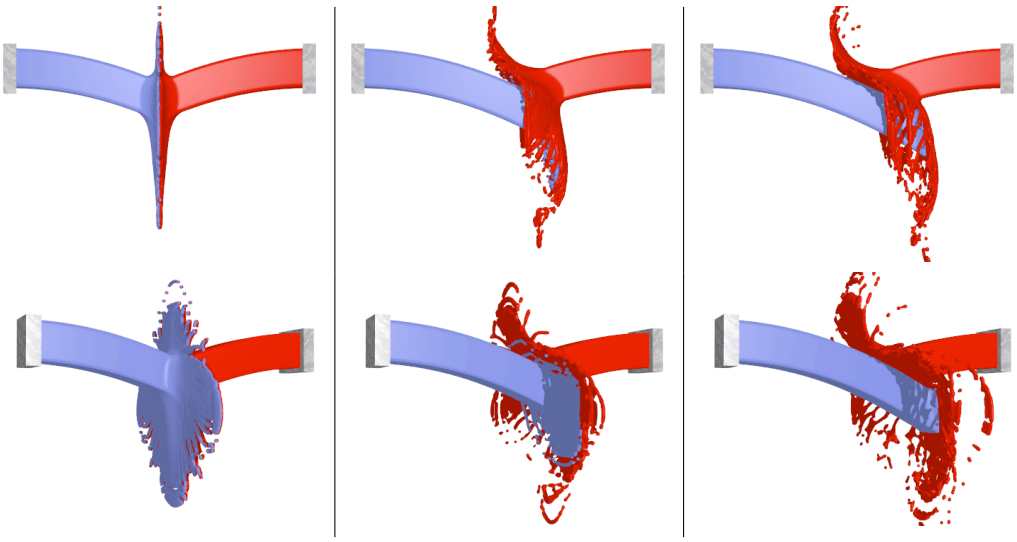


Figure 4.10: *From left to right: two fluids with a density ratio of 1, 10, and 100, respectively.*

CHAPTER 5

UNIFIED PARTICLE MODEL

5.1 Fluid-Solid Interactions

The interaction between fluids, solids, and their surroundings is becoming increasingly important in computer graphics. These interaction processes are physically very complex and very difficult to simulate. It is highly desirable to have a single simulation method which can handle different types of materials, and interactions between them as well. This eliminates the need to define an interface for coupling different fluid and solid models. Currently, coupled models are widely used in computer graphics, but the variety of the simulated materials and effects is often constrained by the interfaces between the models (Table 5.1). Furthermore, they suffer from the effect, that their methods, or the combination thereof, are not appropriate to simulate the whole variety of interaction processes, including phase changes between fluid and solid. Especially melting and solidification which are caused by a surrounding liquid did not receive enough attention yet, although they contribute a lot to a realistic simulation of the interaction of fluids with their environment.

Lagrangian mesh-based and mesh-free methods are widely used in computer graphics for the simulation of deformable objects as well as for fluids. With the preliminary work of [Terzopoulos et al., 1989], melting of solids into fluids through heat and heat transfer using particle dynamics was introduced in graphics. Their deformable objects are represented by mass-spring systems (MSS) and melting is achieved by varying the spring constants and finally removing the

	[Tonnesen, 1991]	[Baraff, 1997]	[Carlson et al., 2002]	[Müller et al., 2003]
Rigids		x		
Elastics	+			
Rigid-Elastics[1]				
Fluids	x		x	x
Melting	x		+	
Solidification	+		+	
Distinction[2]		–		
Merging[3]	+		x	
Splitting[3]	+		x	

	[Carlson et al., 2004]	[Müller et al., 2004]	[Keiser et al., 2005]	[Losasso et al., 2006a]
Rigids	x			x
Elastics		x	x	
Rigid-Elastics[1]				
Fluids	x	x	x	x
Melting		x	x	x
Solidification		+	+	
Distinction[2]	–		x	–
Merging[3]		x		
Splitting[3]		x		

Table 5.1: *Overall comparison of the effects to related previous work. Our model can handle all effects listed above.*
x: Effect is covered in previous work
+: Effect is covered in previous work but with limitations
–: not applicable
[1]*: objects consisting of rigid as well as elastic parts*
[2]*: between multiple close (touching) deformable objects and parts of the same deformable object which are close (touching)*
[3]*: as illustrated in Figure 5.1*

springs. The liquefied particles then interact with Lennard-Jones (LJ) potentials, corresponding to a fluid simulation on the microscopic level. This approach is extended in [Tonnesen, 1991], where the MSS is replaced by different LJ potential energy functions that vary the strength of the attractive and repulsive forces to produce fluid or solid behavior according to the particle temperature. Although both previous works succeeded in melting objects, the identification of the relevant parameters of the LJ interaction forces and the MSS remains a major problem. As discussed in [Nealen et al., 2005], spring constants of a MSS are often chosen arbitrarily since the model does not allow the direct integration of physical parameters. This leads to problems when changing the model resolution as it is not

clear how the parameters have to be modified to retain the same behavior. Similar problems exist for LJ potential functions. Another difficulty of MSS is that the behavior of the model is highly dependent on the topology, which is problematic during a solidification process where springs have to be added continuously. The use of different LJ potential functions causes problems during solidification as well, since the equilibrium between gravitational forces and inter-particle forces is shifted, leading to spurious particle expansion or contraction. Using a mesh-free continuum-mechanics-based framework for the animation of elastic objects, as we use in our unified SPH model, offers the advantage of not having to take care about topology at all and that the resolution has only an effect on the accuracy of the method and not on the parameters defining the material properties.

The SPH method was originally developed to model cosmological fluids [Gingold and Monaghan, 1977; Monaghan, 1992] and was introduced to computer graphics in [Stam and Fiume, 1995]. Later, [Desbrun and Cani, 1996] used SPH for the animation of highly deformable objects, and extended it in [Stora et al., 1999] to animate lava by coupling the viscosity to temperature. Since then, SPH has been used for a wide range of applications in computer graphics. [Müller et al., 2003] use SPH for the simulation of fluids at interactive rates. Their work has been extended later to simulate the interaction of fluids with deformable meshes by adding boundary particles to the surface of the mesh [Müller et al., 2004]. The interaction between multiple SPH fluids with different physical properties is introduced in [Müller et al., 2005b]. Melting and freezing using SPH particles is addressed in [Wicke et al., 2006], where particles are subject to elastic restoring forces arranging them in a locally defined lattice. [Müller et al., 2004] proposed a technique to model elastic, plastic and melting behavior of objects using particles, where a Moving Least Squares (MLS) approach is used to calculate the elastic forces. The elastic model is extended in [Keiser et al., 2005], where additionally a method for the handling of topological changes is proposed. [Cani and Desbrun, 1997] presented a method that uses implicit surfaces for animating deformable models. Their elastic objects can collide under low pressure and merge to one object otherwise.

Recent work on the simulation of fluids with Eulerian approaches addressed the simulation of different materials and phase changes, as well as interaction processes between fluids and solids. [Losasso et al., 2006b] presented the simulation of complex interactions between multiple fluids with different physical properties. Two-way interaction between fluid and solid was introduced in [Carlson et al., 2004], where the rigid objects are treated as a fluid constrained to rigid body motion. The coupling between an Eulerian fluid solver and deformable solids was demonstrated in [Génevaux et al., 2003] and [Chentanez et al., 2006], and coupling water to thin deformable and rigid shells was shown in [Guendelman et al., 2005]. A simulation of melting has been presented in [Carlson et al., 2002],

where deformable bodies are represented as a very viscous fluid. Melting is made possible by adapting the viscosity depending on the temperature. By adding elasticity to an Eulerian fluid simulation instead of increasing the viscosity, [Goktekin et al., 2004] achieved animations of viscoleastic fluids. Recently, [Losasso et al., 2006a] introduced a fluid model coupled with a solid simulator, where the solid objects are represented by meshes. Their simulation can handle the melting and burning of solid objects into liquids and gases, but the solidification process is still a major challenge. Using the lattice Boltzmann method (LBM) [Zhao et al., 2006] demonstrated melting and flowing in a multiphase environment.

We refer to an extensive survey on physically based deformable models in [Nealen et al., 2005]. Concerning the dynamics of rigid bodies a comprehensive introduction is given in the notes of [Baraff, 1997].

The model we propose in this thesis is a Lagrangian approach, which we find to be advantageous for the simulation of mixing processes between different fluids and solids. Our work has been motivated by the fact that previous particle models only fulfill a subset of the desired interaction processes as summarized in Table 5.1, which makes it difficult to combine them into a single model. Our approach borrows from many prior particle methods, and thus is not fundamentally different from these, but we have enhanced and altered many critical components as described mainly in the Sections 'Elastic Bodies' and 'Rigid Bodies'. However, the main contribution is the integration of all the presented modifications and effects into a single unified particle model. In our model, fluids and solids are both represented by particles, each of which knows its own attribute values describing its physical properties of matter. Since each particle interacts with its neighboring particles regardless of the state of matter, we achieve a two-way coupled fluid-solid interaction without any further treatment.

Phase change behavior is already addressed in previous work, but the proposed models are limited in the resulting interaction effects, as can be seen in Table 5.1. By using our technique the simulation of a wide range of effects is made possible which are not producible with any single of the previous methods alone. Our model can combine the following properties:

- *Flexibility of materials*: Support for fluids, elastic and rigid objects, and even the combination of both, i.e. elastic and rigid parts, in one single object.

- *Melting and solidification*: A solid body can turn into a fluid when heat is applied and vice versa for cooling. The simulation can handle partial and continuous melting and solidification, even while interacting with a surrounding liquid.

- *Distinction of objects*: The model supports distinction between multiple

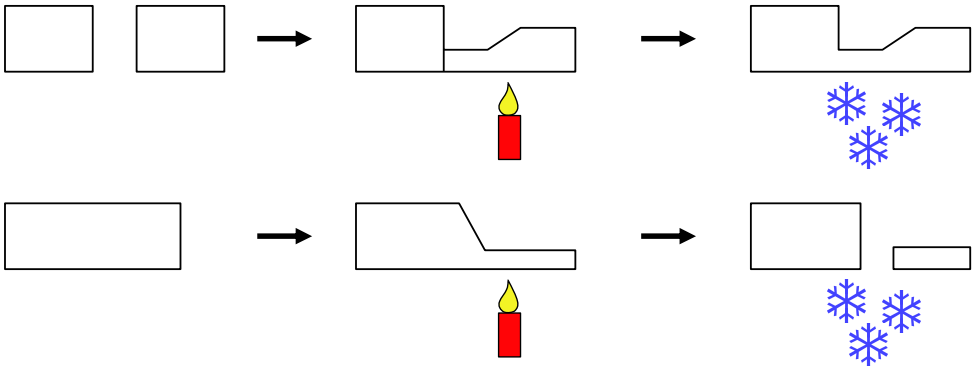


Figure 5.1: *Top: Merging. A solid melts due to heating and touches the other solid in the process. After cooling, the two objects merge to a single one. Bottom: Splitting. A solid melts due to heating and some parts separate. Cooling down leads to two independent solids.*

objects or parts of the same object which are close (touching), as long as no melting is involved in the process.

- *Merging and Splitting*: Ability to merge multiple close (touching) objects into one when melting is involved (Figure 5.1, top) and split objects (Figure 5.1, bottom) as a result of phase changes.

5.2 Elastic Bodies

5.2.1 Model Extensions

Our method for modeling deformable bodies extends the work of [Müller et al., 2004; Keiser et al., 2005], where at every particle position the gradient of displacement from the undeformed (reference) shape of the body is used to compute the strain ϵ, stress σ, and elastic forces $\mathbf{F}^{elastic}$. However, the approach has been altered significantly:

1. In our model we use an SPH approach instead of MLS, which has the advantage that it can handle coarsely sampled and even coplanar particle configurations, as they often result from phase change processes (Figure 5.2). The use of SPH affects Equations (5.5) and (5.7) in the following section.

2. We have modified the elasticity model fundamentally with a new definition of the reference shape of a body. Instead of referring to an initial, global

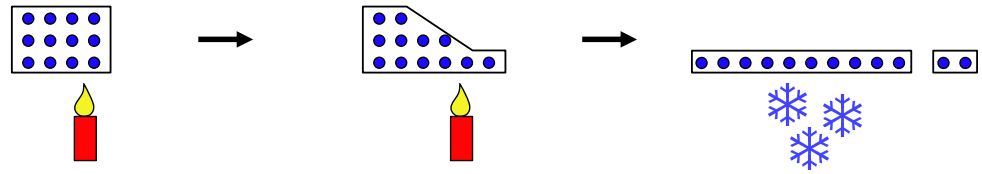


Figure 5.2: *A phase change process may result in coarsely sampled regions and coplanar particle configurations which can be handled by SPH but not by MLS.*

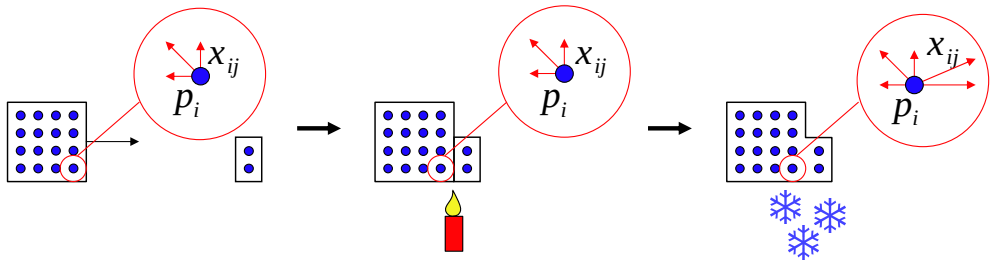


Figure 5.3: *Our model uses a locally undeformed object condition, where each particle stores a distance vector to each of its local neighbors. If objects merge, the reference distances of the new neighbors are added to the particle.*

undeformed reference shape as in [Keiser et al., 2005], we consider a *locally undeformed object* condition. So, instead of storing the position of the reference shape, each particle stores a distance vector to each of its local neighbors. The neighborhood of a particle is defined by the support radius of the SPH smoothing kernel. The locally undeformed object condition is also critical for the merging of multiple bodies into one, and the splitting of bodies as a result of phase changes. This behavior is illustrated in Figure 5.3. Without our extension, if two separated and undeformed bodies move and merge during the simulation, large strains will erroneously be measured. The definition of this local reference shape of a body has effects on the computation of the body volume of a particle (see Equation (5.2)) and on Equation (5.6).

3. In contrast to [Müller et al., 2004], the reference neighborhood of a particle does not change during elastoplastic processes. This property allows for the distinction between multiple close (touching) objects or parts of the same object.

5.2.2 Resulting Elasticity Model

For calculating the elastic force of particle i, we need to determine the strain energy U_i of the particle. This is usually measured in terms of an energy density and is given as

$$U_i = \overline{V}_i \frac{1}{2}(\epsilon_i \cdot \sigma_i), \tag{5.1}$$

where $\overline{V}_i$ is the body volume (reference volume) of particle i. $\overline{V}_i$ is computed as if all of its body neighbors j_{body} were located at a relative position $\mathbf{x}_{ij}$, which is equal to the distance vector between the reference positions of particle i and j in our *locally* undeformed object definition. As the volume is calculated by dividing mass by density the body volume of particle i is defined as

$$\overline{V}_i = \overline{V}(\mathbf{x}_i) = \frac{m_i}{\sum_{j_{body}} m_j W(\mathbf{x}_{ij}, h)}. \tag{5.2}$$

Since we use a linear stress-strain relationship it holds that $\sigma = \mathbf{C}\epsilon$, which is known as Hooke's law. $\mathbf{C}$ is a rank four tensor, approximating the constitutive law of the material, and both ϵ and σ are symmetric 3x3 tensors ([Müller et al., 2004]). In addition, we only use isotropic materials in our simulations, which means that $\mathbf{C}$ depends only on the Young's Modulus E and the Poisson's Ratio ν. There are different formulas for calculating the strain, the one employed here is called the Green-Saint-Venant strain tensor, but it can be easily replaced by a different one.

The elastic force $\mathbf{F}^{elastic}$ can then be defined as the negative gradient of strain energy U with respect to displacement. The force that particle i exerts on its jth neighbor is given by

$$\mathbf{F}_{ji}^{elastic} = -\nabla_{\mathbf{u}_j} U_i = -2\overline{V}_i(\mathbf{I} + \nabla \mathbf{u}_i^T)\sigma_i \mathbf{d}_{ij}, \tag{5.3}$$

where $\mathbf{I}$ is the identity matrix, $\nabla \mathbf{u}_i$ is the gradient of the displacement from the reference shape of the body and $\mathbf{d}_{ij}$ is defined by

$$\mathbf{d}_{ij} = \frac{\partial \nabla \mathbf{u}_i}{\partial \mathbf{u}_j}. \tag{5.4}$$

For more detailed derivations of (5.3) and (5.4), refer to [Müller et al., 2004]. We calculate (5.4) using the SPH method, thus $\nabla \mathbf{u}$ is defined as

$$\nabla \mathbf{u}_i = \sum_{j_{body}} \overline{V}_j \nabla W(\mathbf{x}_{ij}, h)(\mathbf{u}_{ji})^T, \tag{5.5}$$

where the displacement difference vector $\mathbf{u}_{ji}$ is a function of the current position $\mathbf{r}$ and $\mathbf{x}_{ij}$:

$$\mathbf{u}_{ji} = \mathbf{u}_j - \mathbf{u}_i = \mathbf{r}_j - \mathbf{r}_i + \mathbf{x}_{ij}. \tag{5.6}$$

The derivative of (5.5) with respect to $\mathbf{u}_j$ is computed by the resulting SPH equation for $\mathbf{d}_{ij}$

$$j \neq i \rightarrow \mathbf{d}_{ij} = \overline{V}_j \nabla W(\mathbf{x}_{ij}, h). \tag{5.7}$$

5.2.3 Elasticity Kernel

The kernel function used to calculate a certain attribute or force has a great influence on the behavior of an SPH simulation. The most obvious effects of the smoothing kernel are those on stability and speed. When melting and solidification is introduced, arbitrary sets of particles can solidify into deformable bodies. While a fluid cools, there may be parts where already very few cold particles (possibly only two) form small objects. The "spiky kernel" presented in [Desbrun and Cani, 1996] turned out to be unable to cope with such situations since the simulation becomes unstable as soon as elastic forces are computed for particles with a deficient neighborhood. To cope with this problem, we replaced the spiky kernel with a sine shaped kernel function (Figure 5.4):

$$W(\mathbf{r}, h) = \begin{cases} c\frac{2h}{\pi}\cos(\frac{(r+h)\pi}{2h}) + c\frac{2h}{\pi} & 0 \leq r \leq h \\ 0 & \text{otherwise.} \end{cases}$$

As we use W in a normalized form, c is determined by

$$c = \frac{\pi}{8h^4(\frac{\pi}{3} - \frac{8}{\pi} + \frac{16}{\pi^2})}. \tag{5.8}$$

Given that c is a constant, it can be precomputed at the beginning of the simulation.

5.2.4 Plasticity and Fracture

Objects in the real world are not perfectly elastic. Depending on the amount of experienced strain, materials often do not fully return to their original shape. This effect is called plasticity, which we capture by integrating the model proposed in [O'Brien et al., 2002]. If plastic flow occurs in an object, a part of the deformation is absorbed by the material, and its shape is permanently changed. However, this occurs only after the object has been deformed sufficiently, which can be defined by an elastic limit. We use the von Mise's yield criterion which is based on the deviation of the elastic strain as described in [O'Brien et al., 2002]. We first compute the base change of plastic deformation as well as the plastic strain. Then, the current elastic strain of each particle is then represented by the difference between the particle's plastic strain and the particle's total strain. Moreover, the plastic deformation will not go beyond some plastic limit. Every particle knows

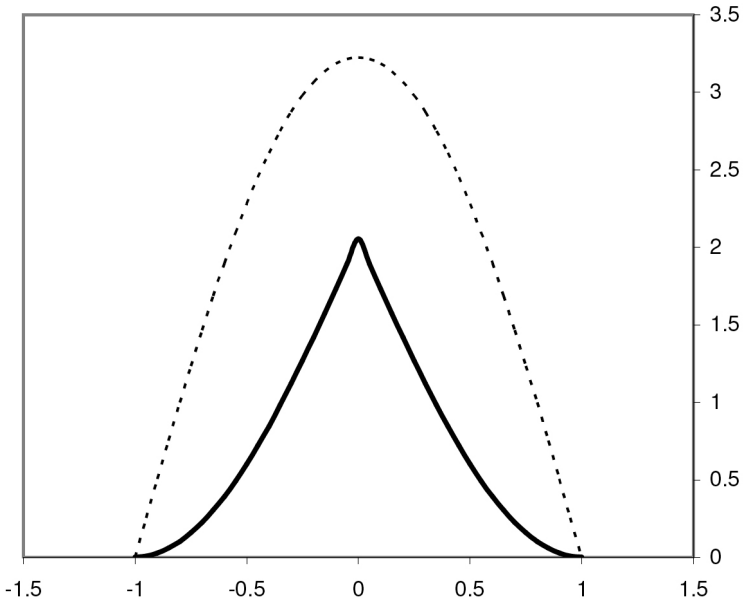


Figure 5.4: *The smoothing kernel along one axis used for the elastic forces, for smoothing length h=1. The thick line shows the kernel and the dotted line its gradient.*

its own elastic and plastic limits, which makes it possible to simulate different materials at the same time.

Additionally, a simplified fracture rule is added to the deformable model, considering the distance between two body neighbors. If this distance gets too large and exceeds a limit, they discard each other as object neighbors. As a result, they do not perform any forces on each other anymore and the object breaks at this location.

5.3 Rigid Bodies

For modeling rigid bodies the basic SPH fluid model is extended to enforce rigid body motion. For that, the total forces acting on the particles belonging to a rigid body are accumulated in the body, then its movement is restricted to translation and rotation [Baraff, 1997]. The rigidity method described here builds upon the body neighbor object representation introduced for elastic objects. Our model keeps track of particles that belong to the same rigid body to allow for merging and splitting during phase change processes as presented in the following section.

In order to constrain the motion of an object to rigid body motion, we have to handle rotation explicitly. To do so, we compute a torque vector τ according

to [Baraff, 1997]:

$$\tau_i = (\mathbf{r}_i - \mathbf{r}^{cm}) \times \mathbf{F}_i, \tag{5.9}$$

where $\mathbf{r}^{cm}$ is the center of mass of a body and $\mathbf{F}_i$ denotes the total force exerted on the ith particle. $\mathbf{F}_i$ is the sum of all forces calculated with SPH (the force densities $\mathbf{f}_i$ are multiplied by the volume of i to get forces) and all external forces present in the simulation. The total force acting on a body is given by $\mathbf{F}_{body} = \sum_{i_{body}} \mathbf{F}_i$ and the total torque is defined by $\tau_{body} = \sum_{i_{body}} \tau_i$. Note that for efficiency, the computation of force densities between pairs of particles that belong to the *same* rigid body are skipped.

After the total force and torque of a body are determined, time integration is performed by first iterating over a rigid object to calculate the effect of the forces and torques, i.e. the effect on the position and the linear and angular velocity of the body, then the particles belonging to the body are updated to reflect the changes of their parent. The angular velocity of a rigid body is defined as

$$\omega = \mathbf{I}^{-1}\mathbf{L}, \tag{5.10}$$

where $\mathbf{I}$ is the inertia tensor and $\mathbf{L}$ is the angular momentum, which is updated in every time step by calculating

$$\mathbf{L} \leftarrow \mathbf{L} + \tau \Delta t. \tag{5.11}$$

5.4 Phase Changes

5.4.1 Temperature Effects

A temperature is attributed to every particle. It can change either because of heat diffusion among neighbor particles or because of outside influences. Using the SPH formalism, the evolution of the temperature T_i due to diffusion sampled at the particles can be computed analogously to [Stora et al., 1999] as

$$\frac{\partial T_i}{\partial t} = c \sum_j m_j \frac{T_j - T_i}{\rho_j} \nabla^2 W(\mathbf{r}_{ij}, h), \tag{5.12}$$

where T is the temperature and c is a diffusion constant. We integrate the attributes in time using a simple Euler-scheme. Additionally, every particle stores a melting point t_{melt} and a solidification point t_{solid} according to the material. If t is above t_{melt}, the particle is liquid, and if t is below t_{solid}, it is solid, belonging either to a deformable body with maximal Young's Modulus E (maximal elastic stiffness) or to a rigid body. In between, the particle belongs to a deformable body with E and the viscosity μ interpolated linearly, whereas the Poisson's Ratio ν stays

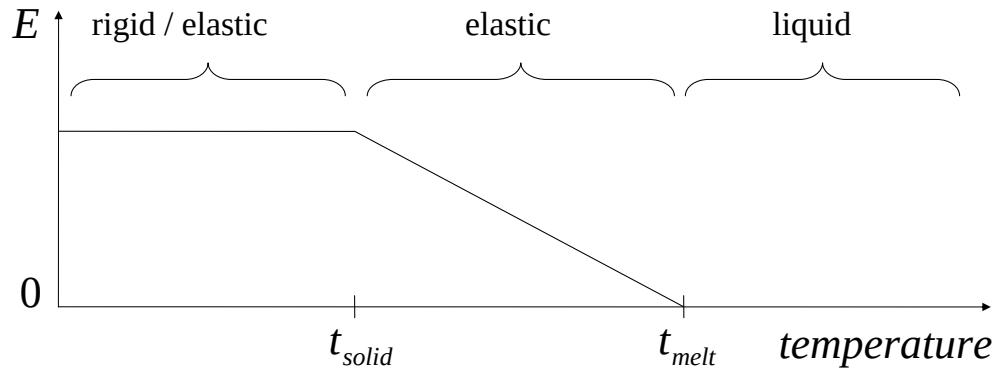


Figure 5.5: *Stages of the phase change process. A particle belongs either to a liquid, an elastic or a rigid object according to its type, its current temperature and its melting and solidification temperatures stored.*

constant (Figure 5.5). By choosing $t_{solid} = t_{melt}$ the intermediate state is left out and it is possible for a liquid to solidify directly to a rigid body or melt a rigid body directly into a liquid.

5.4.2 Phase Changes of Elastic and Rigid Bodies

During melting, a particle must be able to separate from its parent object as soon as it is liquified, and it must be able to merge with a touching solid object during solidification as well. To model this behavior for *elastic* particles the following needs to be updated: the set of body neighbors j_{body}, the set of reference distance vectors $\mathbf{x}_{ij}$, and the body volume $\overline{v}_i$. Note, this has to be done for both the particle that solidifies or melts, and the body neighbors that are added or discarded respectively. For merging and splitting behavior of *rigid* objects, the model must keep track of particles that belong to the same rigid body. In all these cases we have to update the following object quantities: the mass m, the center of mass $\mathbf{r}^{cm}$, the inertia tensor $\mathbf{I}$, the velocity $\mathbf{v}$ and the angular momentum $\mathbf{L}$.

5.5 Results

We have tested our method with several example simulations. The aim of these examples is not to compete for highest visual quality but to demonstrate the flexibility of the unified model, and to show the wide range of interaction effects not producible with any single of the previous methods alone. In Table 5.1 the effects covered by our model are compared to previous work which is related to or

integrated in our method.

All simulation scenes are performed with about 40,000 particles (except the scene in Figure 5.6 which consists of 3,000 particles) on a Intel iMac 2GHz. Note that the performance measurements shown below cannot be directly compared to the timings of Chapters 3 and 4 since different hardware has been used for the computations.

In our simulation, the calculation of the physics takes around 0.5 second per frame in the slowest case, whereas high quality raytracing using our surface reconstruction including shadows and antialiasing take together on average 1 minute per frame in Povray for a resolution of 640x480.

Figure 5.6 shows the rigid-elastic interaction on the left and the elastic-elastic interaction on the right. Apart from choosing a high gas constant k, no special collision handling is necessary to avoid penetration of the objects.

After heating the ground, the rigid (blue) and elastic (red) blocks melt until cooling is turned on in Figure 5.7. This leads to solidification and merging of all objects since they touch each other. The resulting object consists of rigid as well as elastic parts.

Figure 5.8 shows a grey block which is moved on a board. While passing the other blocks, heating is in some cases turned on. Due to heating and cooling, the first two blocks melt at one side and merge to a single solid afterwards. If there is no heating involved as with the last two blocks they do not stick to the moving block although they touch each other as it moves by.

In Figure 5.9 a rigid bunny is dropped into a liquid and starts to melt as soon as the temperature exceeds its melting point. Slower melting can be realized by choosing a higher melting point or by reducing the temperature diffusion rate. Different buoyancy behavior can be achieved by varying the density of the bunny particles.

In Figures 5.10 and 5.11 hot liquid matter is dropped into a cold viscous liquid, i.e. cream. Due to temperature diffusion, the poured liquid cools down and either fully (Figure 5.10) or partially (Figure 5.11) solidifies to a shell. The partially solidified shell is lifted up and rotated while the simulation of the cream is temporarily stopped. Due to gravity, the still hot and liquid part of the shell flows out and solidifies after colliding with the cold splash of the cream. The color shows the object temperature, where a light color corresponds to a high temperature and a dark to a low one.

A solid cold white chocolate bunny is dipped into hot brown chocolate in Figure 5.12. Depending on the time in the chocolate, the bunny's head gets coated with it (dark brown color indicates that the chocolate is cold) or melts.

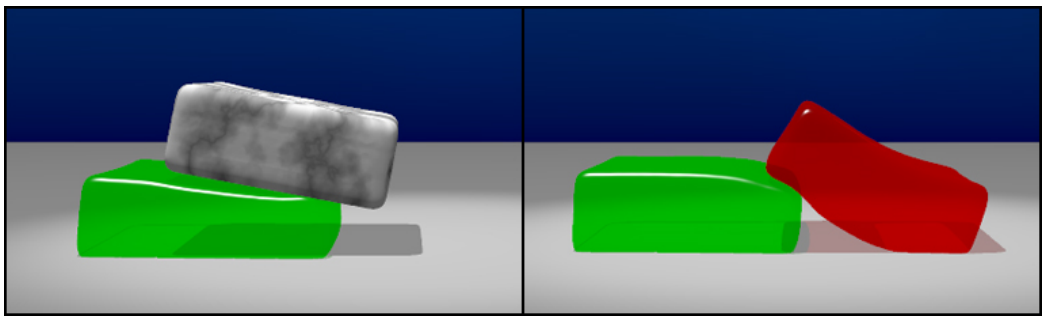


Figure 5.6: *Rigid-elastic interaction (left) and elastic-elastic interaction (right). No special collision handling is necessary to avoid penetration.*

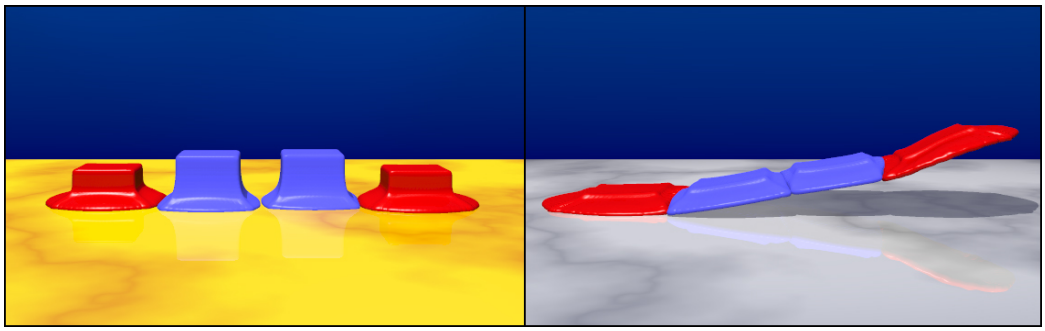


Figure 5.7: *Merging of rigid (blue) and elastic objects (red) after melting and solidification.*

5.6 Discussion

The scenes in the screenshots do not compete for highest visual quality, since only a few thousand particles are used. Although it is our intention in the future to simulate larger scenes to improve the visual quality, it was the focus to create a model capable of running at interactive rates. Our current physics implementation runs at roughly 2 frames per second using 40,000 particles even without applying elaborated optimization techniques. We believe that an acceleration by a factor of 10 is feasible using a combination of fast incremental neighbor search, parallelism and hardware accelerated techniques. An additional increase in efficiency can be achieved by using adaptive particle sizes to have more details where necessary and less computational costs in unchanging regions.

Open problems exist in the handling of collisions with solids. Particle penetrations may occur if strong forces are involved. In our experience, most collision problems can be handled very well by choosing a high gas constant (Fig-

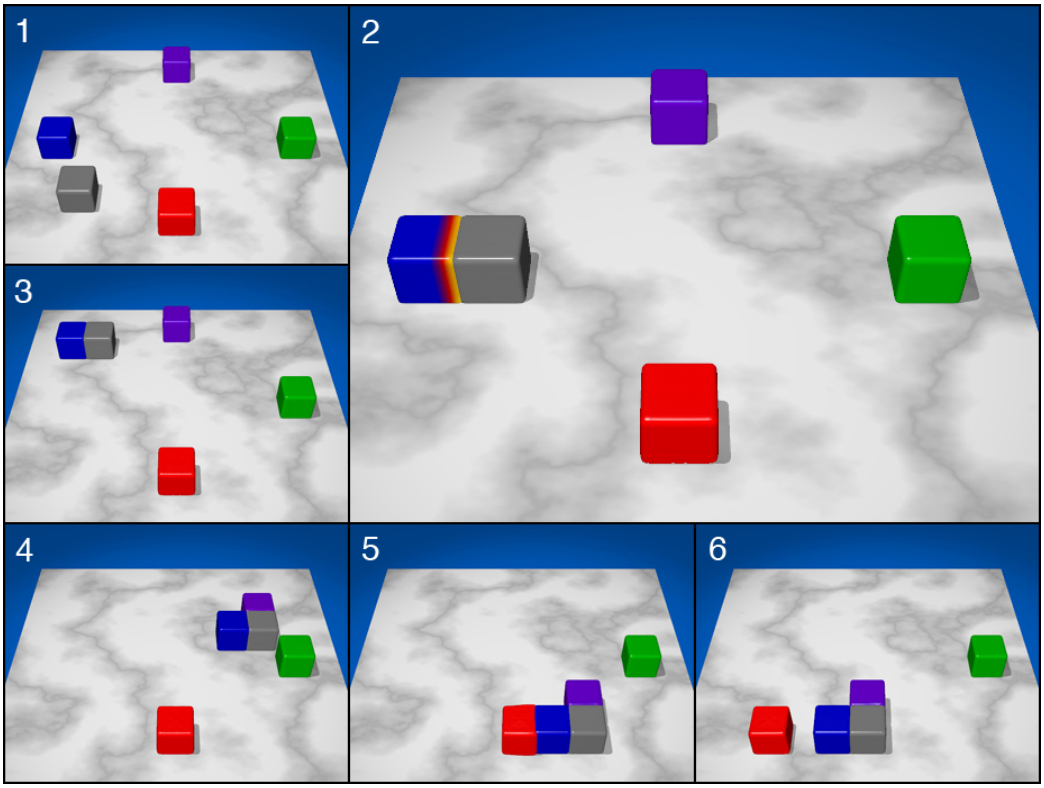


Figure 5.8: *A couple of rigid blocks and one elastic block (red) on a plate. The grey block is moved, passing by the other blocks. If heating is on while passing, the objects merge (blue and purple), otherwise distinction can be observed.*

ure 5.6, 5.8), but an explicit collision handling or additional boundary forces would be desired to guarantee no penetrations despite strong forces involved.

Still to be investigated is how the accuracy of the method relates to the number of particles discretizing a certain amount of volume as well as to the time step used in the Leap-Frog integration scheme. It is difficult to give an estimate of the accuracy of our method. Clear is, that the method gets more accurate by choosing smaller particle sizes and smaller time steps. Further investigations and comparisons with the real world have to be done.

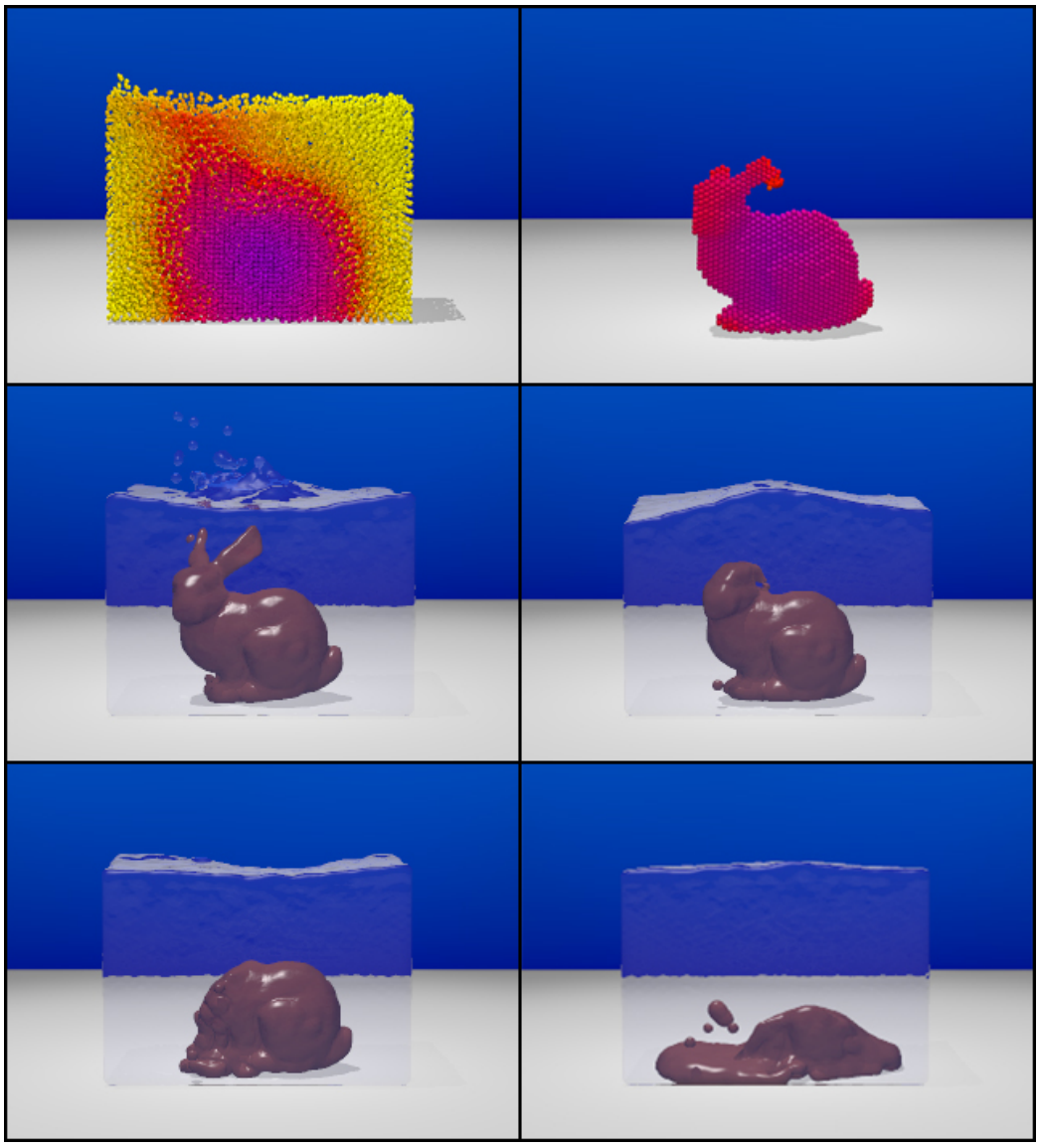


Figure 5.9: *A solid bunny melts inside hot liquid. Top: particles with a temperature coded color, where the left image shows a cut through the particles. Middle, Bottom: raytraced particles.*

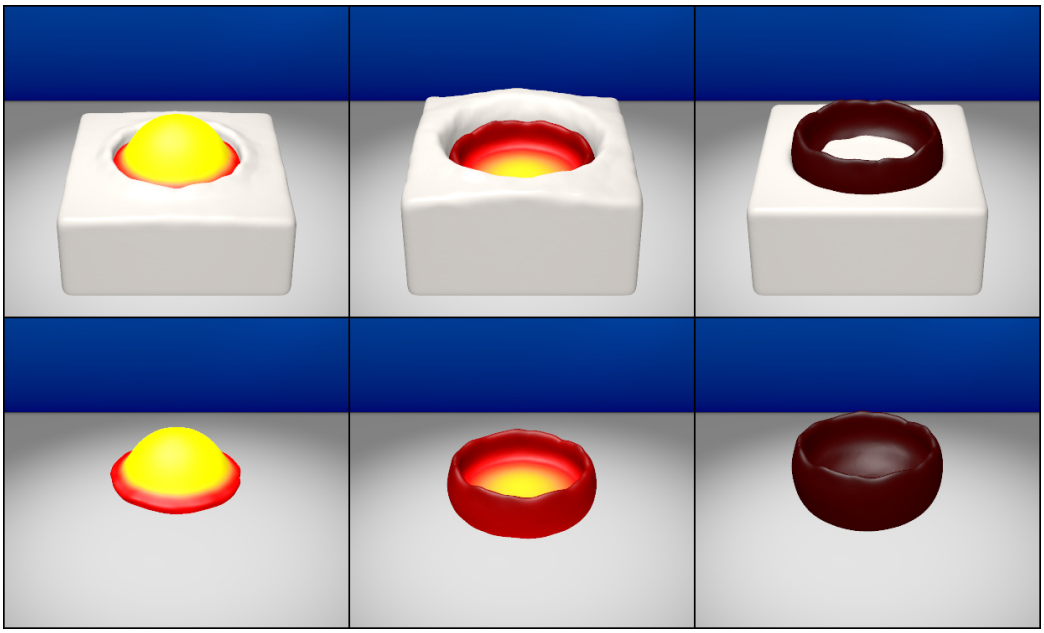


Figure 5.10: *A shell is formed as hot fluid solidifies when it drops into a cold viscous fluid.*

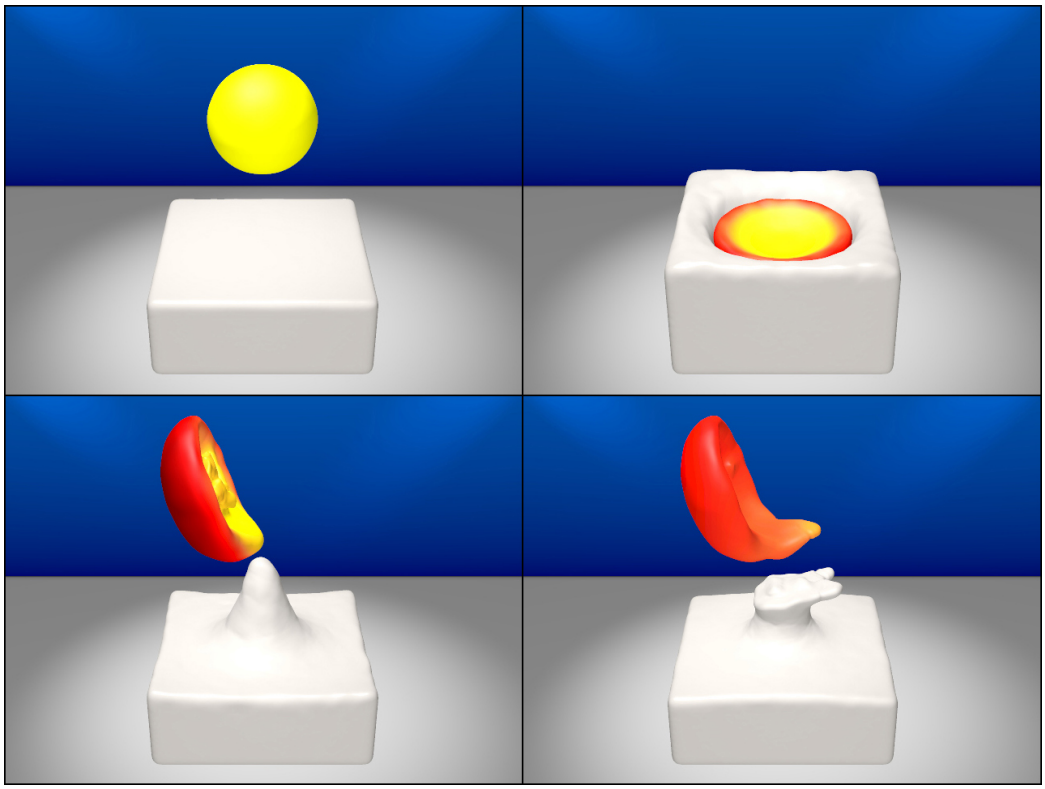


Figure 5.11: *Hot liquid matter partly solidifies inside a cold viscous liquid. After lifting and turning the shell, the liquid inner part flows out and solidifies after colliding with the cold splash.*

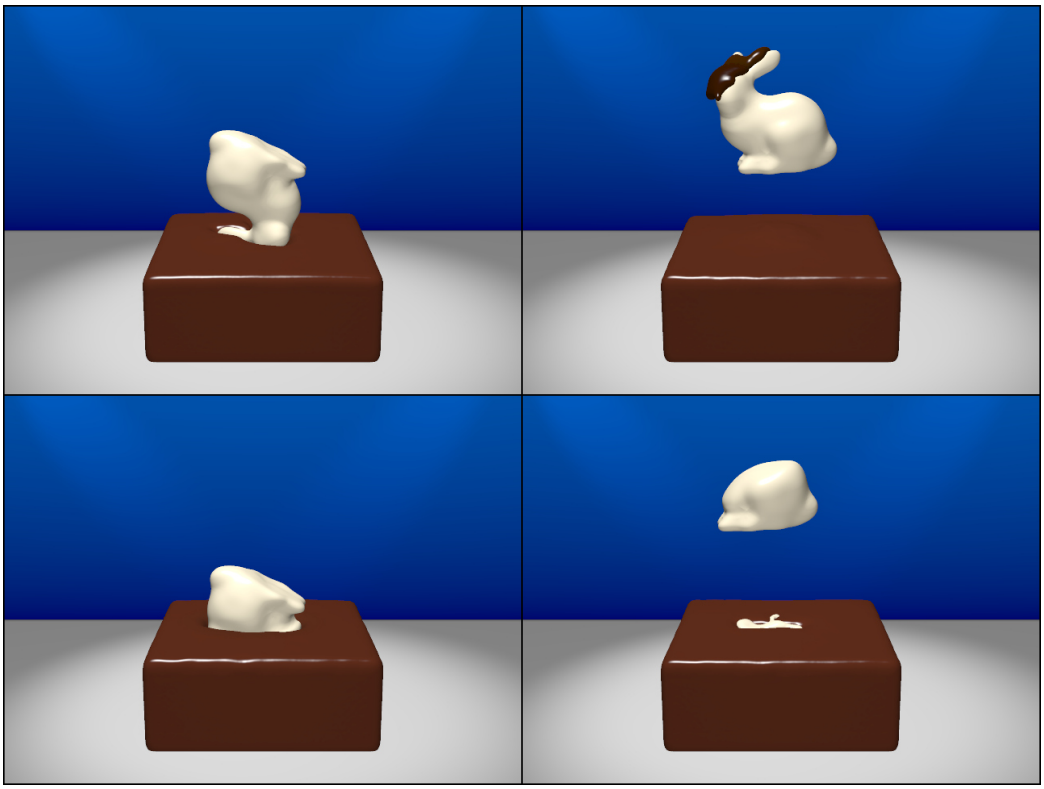


Figure 5.12: *A solid cold white chocolate bunny is dipped into hot brown chocolate. Depending on the time in the hot chocolate the bunny becomes coated with chocolate or melts.*

CHAPTER 6

REFINED SURFACE RECONSTRUCTION

6.1 Smooth Surfaces and Upsampling

Point representations have been used successfully in geometric modeling and in physically based particle systems. The lack of topological and connectivity information simplifies modeling interaction effects (eg. [Müller et al., 2004; Müller et al., 2005a; Müller et al., 2005b; Pauly et al., 2005; Solenthaler et al., 2007a]) as well as geometric manipulations (e.g. [Zwicker et al., 2002; Adams and Dutre, 2003; Pauly et al., 2003; Pauly et al., 2006]). However, it comes at a cost, as neighborhood information has to be computed. Nevertheless, [Müller et al., 2003] succeeded in interactively simulating and rendering particle-based fluids and demonstrated its applicability to virtual reality simulators and 3D games [Müller et al., 2004; Ageia, 2005]. However, due to the real-time constraint, the number of simulated particles has to be low which causes a loss of visual quality. Surface details are smoothed out as a result of surface reconstruction techniques and bumps related to the coarse particle distribution are visible. Using a point splatting approach as rendering technique is also not feasible as under-sampled geometries show artifacts at the silhouette and blur due to large splat radii. It would be desirable to improve the visual quality of low resolution particle simulations while still running at interactive frame rates.

The approach presented in this thesis is to use an upsampling algorithm on the

surface particles to optimize the visual appearance of particle simulations. Such a technique avoids the overhead of running a high resolution physical simulation which is substantial as the number of physical particles increases disproportionally with the desired number of surface particles. Even worse, the Courant condition requires smaller time steps for higher resolutions ([Monaghan, 1989]) resulting in a computational effort of physical simulations increasing quadratically with the number of desired surface particles.

To reconstruct the surface from a set of fluid particles several techniques have been proposed, all without upsampling the surface points of a fluid. An efficient approach is presented in [Müller et al., 2003] where they render the isosurface of a color field defined by the particles. A grid-based level-set simulation guided by particles is presented in [Premoze et al., 2003], where they succeeded in achieving high visual quality but at the expense of computation time. [Zhu and Bridson, 2005] presented a reconstruction technique using an implicit function defined on the center of mass of a local neighborhood of the particles leading to very smooth surfaces. Unfortunately, this method suffers from artifacts in concave regions which they propose to remove in a post processing step. In [Solenthaler et al., 2007a], a method is presented to detect and avoid these errors on the fly. Another extension has been presented in [Adams et al., 2007], where particle-to-surface distances are used for the reconstruction of surfaces from adaptively sized particles.

In geometry processing, a surface reconstruction based on the use of Radial Basis Functions with global support is presented in [Carr et al., 2001], whereas [Ohtake et al., 2003; Tobor et al., 2004] reduce the support by local approaches. MLS surface reconstruction [Levin, 2003] has shown to be successful in surface editing [Pauly et al., 2003], raytracing [Adamson and Alexa, 2003], and up- and down-sampling [Pauly et al., 2002; Alexa et al., 2003]. A sphere fit MLS improving the stability of the projection in low-sampled and curved regions has been presented in [Guennebaud and Gross, 2007]. However, since the projection procedure of the MLS is quite expensive, it is unsuitable for upsampling a set of points in real-time. Real-time upsampling restricted to static, uniformly sampled point data was presented in [Guennebaud et al., 2004]. [Guennebaud et al., 2005] overcomes this weakness and presented a method which is able to fill large holes of static point clouds.

In this thesis, we propose an efficient upsampling method applicable, but not limited to irregularly sampled, dynamic point data in order to reveal all details present in (low resolution) particle-based fluid simulations (for interactive and real-time applications). The main features of our method are:

- **Low computational costs**: No point preprocessing is necessary, new points are added efficiently, and neighborhoods are updated dynamically instead of

being determined from scratch each refinement step, reducing computation and memory costs.

- **Irregular inital sampling**: SPH particles are often irregularly distributed during the simulation. The refinement procedure effectively copes with this problem, and holes are generated only if indicated by the physics simulation.

- **Uniform sampling after refinement**: Point collisions are detected and avoided resulting in a nearly uniform sampling.

- **Details and sharp features**: Our interpolation yields many details of surfaces and splashes, and preserves features like edges.

- **Splashes and isolated particles**: Although isolated particles and particles in splashes posess low-quality normals, curvature and connectivity are preserved in a plausible way.

To achieve a smooth surface from particles, we propose a surface reconstruction similar to [Zhu and Bridson, 2005], but with reduced reconstruction artifacts even for inhomogeneously distributed and sparse particles.

6.2 Refinement

6.2.1 Surface Particle Detection

Since we are interested in visualizing the fluid surface, we only want to refine surface particles. The detection of free surface particles is a difficult problem and has shown to be a critical step in our refinement procedure as erroneously detected and erroneously undetected particles can lead to surface artifacts. In the particle simulation literature, several methods to detect surface particles have been proposed, often based on the number of neighbors in the support radius. These techniques do not satisfy our needs as particles are partly erroneously detected as surface, especially when high pressure forces are involved. This is ascribed to the irregular particle distribution and therefore to the non-constant number of particles in the neighborhoods.

We propose a method to detect surface particles which is based on the distance from the particle to the normalized center of mass $\mathbf{cm}$ of its neighborhood N. This criterion still investigates the particle distribution in the neighborhood of a particle but is independent on the actual number. In our experiments, this technique has proven to be stable in all situations including regions with high pressure forces as

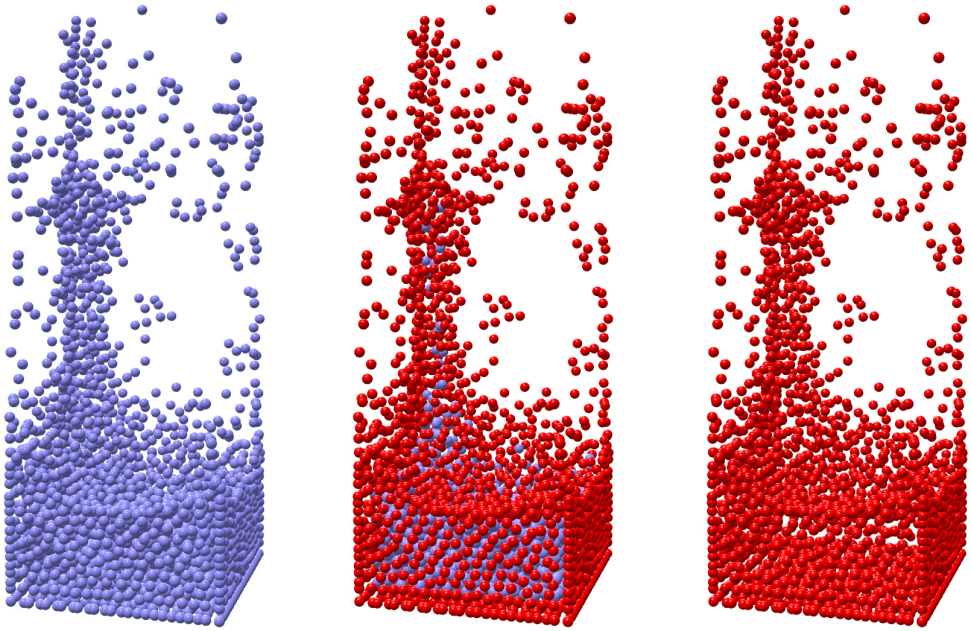


Figure 6.1: *Our surface particle detection method applied to a splashing scenario (left). The red particles are detected as being surface (middle) and correspond to the input set of the upsampling procedure (right).*

well as splashes (Figure 6.1). For each particle p_i, this distance $d_{i,cm}$ is calculated by

$$d_{i,cm} = \frac{\sum_j (\mathbf{x}_i - \mathbf{x}_j) m_j}{\sum_j m_j}. \tag{6.1}$$

A surface particle is defined by having a $d_{i,cm}$ exceeding a certain threshold. In order to avoid oscillations between being surface and not being surface, we use a slow reaction for surface particles to turn into the state of not being surface. This is achieved by using two different thresholds, where a lower one is used for particles which are already marked as surface. Isolated particles have to be treated separately, where they are defined by having an empty neighborhood.

6.2.2 Point-Normal Interpolation

Initial Neighborhood

We use the surface particles as initial point set P_0 for the refinement procedure (Figure 6.1). Similar to [Guennebaud et al., 2004] we insert additional points which yield a new point set P_1 with $P_1 \supset P_0$. The new point set P_1 is then used for the next refinement step. This procedure can be repeated until the desired

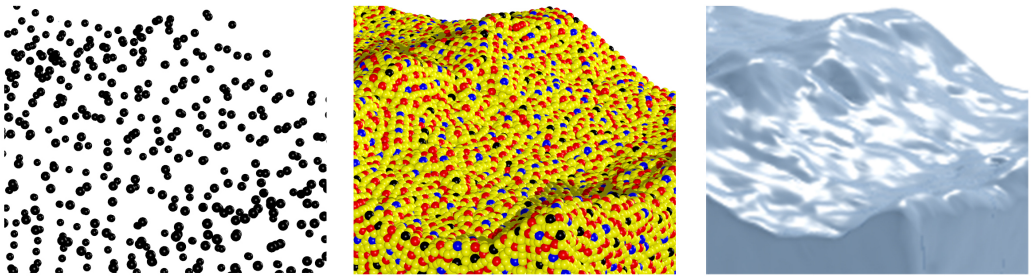


Figure 6.2: *A wavy scene is upsampled from 2.5K to 110K points. The point colors correspond to the different refinement steps: black P_0, blue P_1, red P_2, and yellow P_3.*

point resolution is reached. During one refinement step, an additional point is inserted between each point p_i and its neighbors p_j, where it is avoided to generate points which are too close to points already created in order to achieve a nearly uniform sampling (Section 6.2.3). Figure 6.2 illustrates the points of the individual refinement steps. The neighborhood $N_i^{surface}$ of each point $p_i \in P_0$ is inherited from the physics simulation, where $N_i^{surface} \subset N_i$: if two surface particles are visible to each other in the physics calculation, meaning that they interact with each other, the pair is refined. Otherwise, if two particles have a distance larger than the support radius and do not interact with each other, no additional point is added. That is, the initial visibility radius r_0 used in the refinement procedure is equal to the support radius r used in the physics. Note that r_0 is the same for all points. In the following refinement step, the radius is reduced as described in Section 6.2.2. The point normals are determined either in the physics by using the method proposed in [Müller et al., 2003] or by using a transformation invariant homogeneous covariance analysis as described in [Pajarola, 2003]. The latter is more expensive but leads to higher quality normals which is crucial when using point splatting as rendering technique (see Figure 6.12).

Spherical Interpolation

If a new point is added, its position and normal have to be determined. For real-time applications, it is important that we get these values at low computational costs. Nevertheless, surface details should be preserved as far as possible. We propose a spherical interpolation method which sets a new point (child $c_{1,2}$) onto a sphere which is defined by the two points being refined (parents p_1 and p_2), as illustrated in Figure 6.3. The child point is set onto the perpendicular bisector of $\mathbf{d} = \mathbf{x}_1 - \mathbf{x}_2$, where the displacement from $\mathbf{d}$ onto the sphere is called h. The

position of the child $\mathbf{x}_c$ is given by

$$\mathbf{x}_c = \frac{\mathbf{x}_1 + \mathbf{x}_2}{2} + h\mathbf{n}_c, \tag{6.2}$$

where $\mathbf{n}_c$ is the normal of $c_{1,2}$ as defined in Equation 6.13.

Each parent computes a displacement h_1 and h_2, respectively, and the final h is a function of h_1 and h_2. There are different possibilities to determine h, one is to take the average, another one is to take the absolute minimum:

$$h = \frac{h1 + h2}{2} \tag{6.3}$$

$$h = min(|h1|, |h2|). \tag{6.4}$$

The first one (illustrated in Figure 6.3) leads to a smooth surface whereas the second one preserves edges and corners more accurately. We prefer the second one and our results are all computed using this approach.

To facilitate the calculation of h_1 and h_2, we determine the slopes $\frac{s_1}{t_1}$ and $\frac{s_2}{t_2}$. The displacements are given by

$$h_1 = \frac{s_1}{t_1}\frac{|\mathbf{d}|}{2} \tag{6.5}$$

$$h_2 = \frac{s_2}{t_2}\frac{|\mathbf{d}|}{2}, \tag{6.6}$$

where s and t are defined by

$$s_1 = |\mathbf{n}_1 \cdot \mathbf{a}| \tag{6.7}$$

$$t_1 = \mathbf{n}_1 \cdot \mathbf{b} + \|\mathbf{n}_c\| \tag{6.8}$$

and

$$s_2 = |\mathbf{n}_2 \cdot \mathbf{a}| \tag{6.9}$$

$$t_2 = \mathbf{n}_2 \cdot \mathbf{b} + \|\mathbf{n}_c\|. \tag{6.10}$$

$\mathbf{a}$ and $\mathbf{b}$ are two basis vectors, where the normal $\mathbf{n}_c$ of the child point corresponds to $\mathbf{b}$:

$$\mathbf{a} = \frac{\mathbf{d}}{|\mathbf{d}|} \tag{6.11}$$

$$\mathbf{b}' = \mathbf{n}_1 + \mathbf{n}_2 - ((\mathbf{n}_1 + \mathbf{n}_2) \cdot \mathbf{a})\mathbf{a} \tag{6.12}$$

$$\mathbf{b} = \mathbf{n}_c = \frac{\mathbf{b}'}{|\mathbf{b}'|}. \tag{6.13}$$

$|\mathbf{b}'|$ can be zero if the normals of both parents are parallel to $\mathbf{d}$ or if the normals sum up to zero. In these situations, we assume that the points belong to different surfaces and, therefore, the points are not refined.

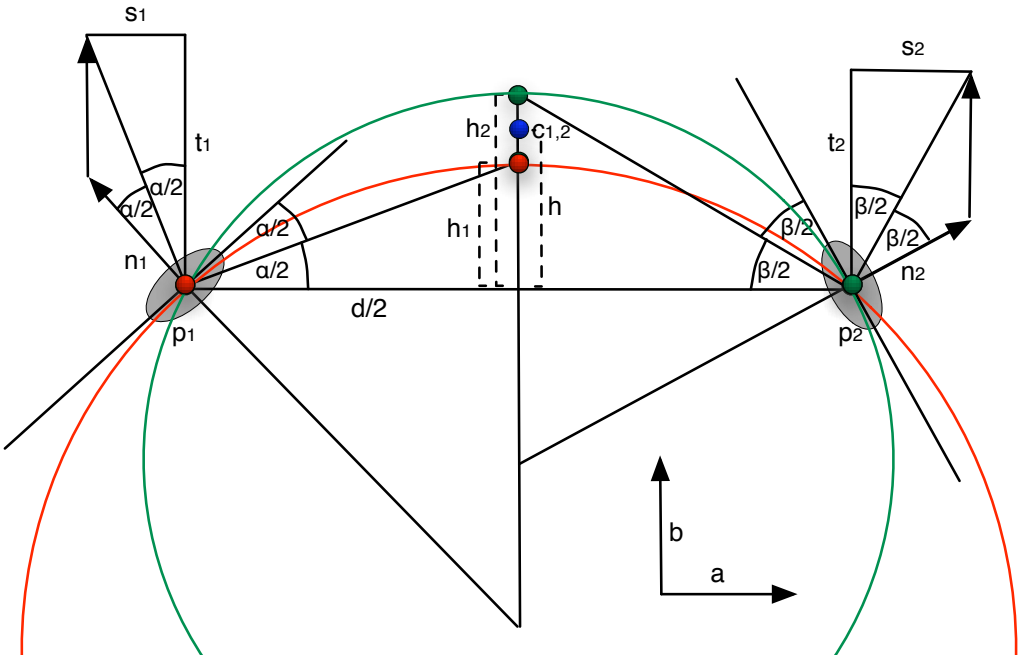


Figure 6.3: *A new child point $c_{1,2}$ is added between the parent points p_1 and p_2 onto a sphere. Each parent computes its $h1$ and $h2$, respectively, where the final h is a function of $h1$ and $h2$.*

Radius in the Next Refinement Step

For efficiency reasons, it is desirable to limit $|h|$ to h_{max}. We chose h_{max} such that 6 points may still refine to a perfect sphere assuming this is a reasonable requirement to the resolution of an underlying simulation. For an illustration in 2d see Figure 6.4.

In this situation, h_{max} is given by

$$h_{max} = \rho - \frac{r_i}{2} = \frac{r_i}{2}(\sqrt{2} - 1), \tag{6.14}$$

where ρ is the sphere radius. As we know h_{max}, the radius r_{i+1} of the next refinement step is always well-defined, since it is required that the child is just in the neighborhood of its parent to avoid parasitic holes. Therefore, r_{i+1} is defined by

$$r_{i+1} = \sqrt{h_{max}^2 + (\frac{r_i}{2})^2} = \frac{r_i}{2}\sqrt{4 - 2\sqrt{2}} \tag{6.15}$$

which means that in each refinement step the radius is reduced by a factor of ≈ 0.54.

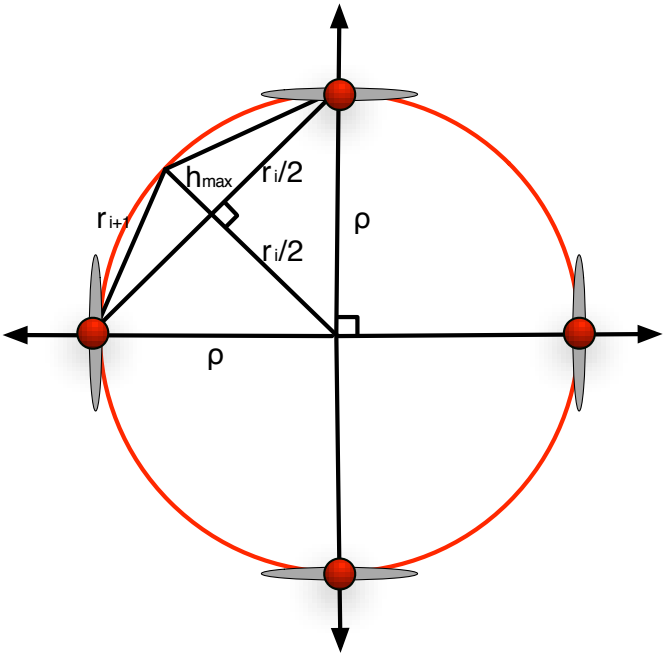


Figure 6.4: *Two-dimensional illustration of the limitation of h to h_{max}. In three dimensions, 6 points are refined to an accurate sphere.*

6.2.3 Point Collision Avoidance

As mentioned above, for each pair of points a new point is inserted. As we want to reach a uniform sampling after the refinement procedure, it is important to avoid adding points too close to existing points.

Collision Detection

We use a collision distance β which defines a sphere around a point which has to be empty. The collision volume of a child point for the case $h = h_{max}$ is two-dimensionally illustrated in Figure 6.5 as red circle. If any other point lies in the same volume the child point is rejected and not added. In each refinement step, β is adjusted proportionally to r_i. We use $\beta = 0.2r_i$ which we have determined heuristically to be adequate to approach a uniform sampling. There are three different collision configurations, see Figure 6.6 for an illustration. (for simplicity reasons, we refer to the different refinement steps as $s_0, s_1, s_2, ..., s_n$, and a point which is added in step s_i is called p^{s_i}):

1. A point created in the current refinement step s_i is closer than β to its parents.

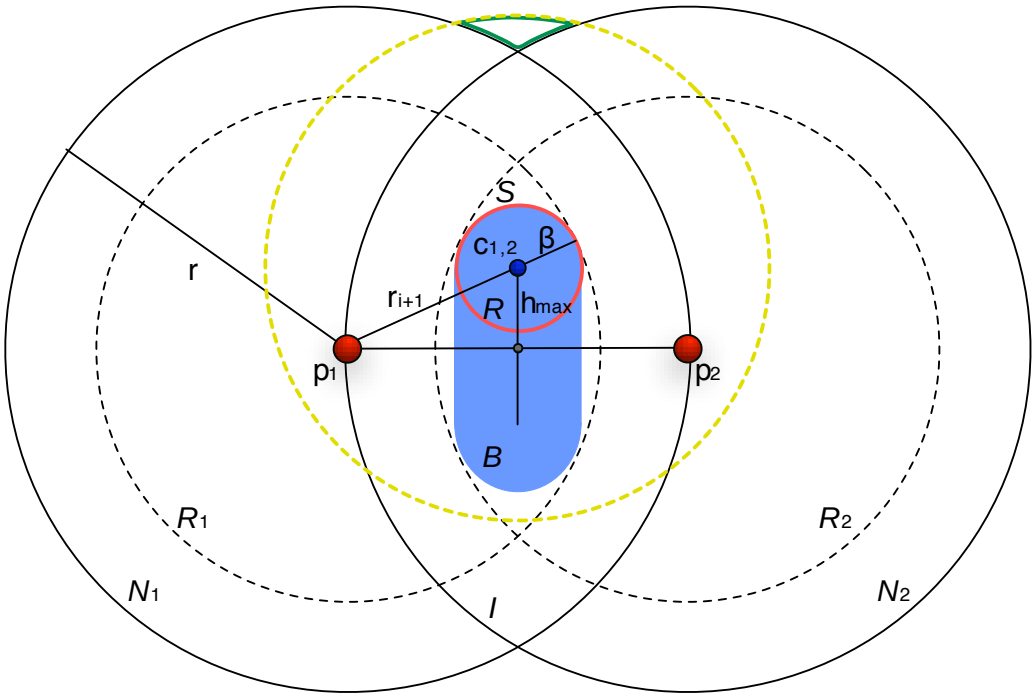


Figure 6.5: *Red: Collision region R for a child point in the case where $h = h_{max}$. Blue: Critical region B which has to be checked for collisions when h is not known. Dashed yellow: A new point p^{s_i} registers at all points $p^{s_{j<i}} \in N_1 \cup N_2$ if point is inside that region (except at points in the green region). Dashed black: R_1, R_2: Registered point sets of p_1 and p_2.*

2. A point created in the current refinement step s_i is closer than β to a (non-parent) point which was created in one of the last refinement steps $s_{j<i}$.

3. A point created in the current refinement step s_i is closer than β to another point which was also created in the current step s_i.

The first type where a child collides with its parents can be avoided by not refining a pair of points which are closer than r_{i+1}. This is valid as long as $\beta < 0.5r_{i+1}$, which is the case when using $\beta = 0.2r_i$. In this situation, the pair gets refined in the next refinement step s_{i+1}.

Since both parents know their neighboring points added in $s_{j<i}$ we can avoid collisions of type 2 by taking the intersection set I of both parent neighborhoods N_1, N_2, which is $I = N_1 \cap N_2$. As can be seen in Figure 6.5, the blue region B corresponds to the critical volume which has to be checked when h is not known. This volume encloses the red region R (collision volume) and is enclosed by the intersection set I: $R \subset B \subset I$. Therefore, all collisions of type 2 can be found by

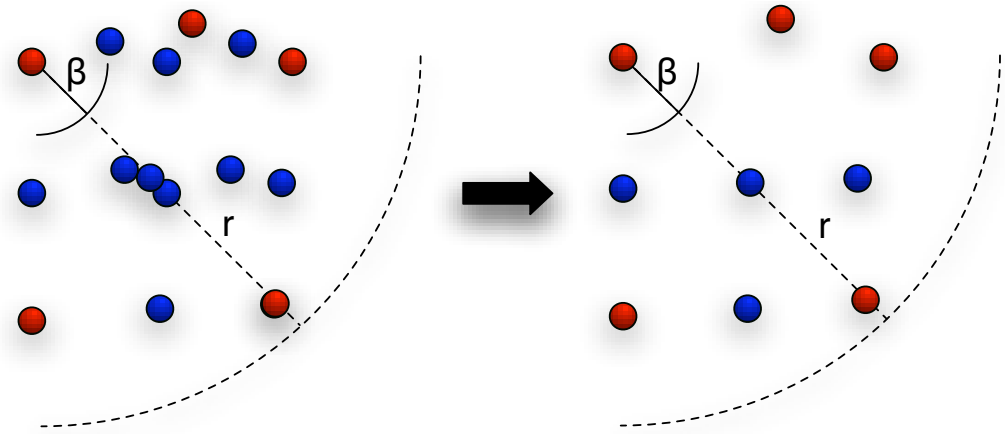


Figure 6.6: *Refined points without and with using our point collision avoidance algorithm. The red points were added in s_0 and the blue points in s_1.*

distance checking the child with all $p^{s_{j<i}} \in I$. Since the number of points in I is relatively small, the fact that I is larger than R is not very time critical.

However, this is different for the third type, since the number of p^{s_i} is growing disproportionally. It turned out that an efficient way to solve type 3 collisions is to register an added child point at all points $p^{s_{j<i}} \in (N_1 \cup N_2)$ which are closer than $r_{i+1}+\beta$. For $c_{1,2}$ in Figure 6.5 these points lie all in the yellow dashed region. As a result, each $p^{s_{j<i}}$ knows the newly added points p^{s_i} inside the radius $r_{i+1} + \beta$. For the parent points p_1 and p_2 these registered point sets are illustrated by the black dashed regions R_1 and R_2. To check for case three collisions, we investigate the distances of each point in the intersection set S of R_1 and R_2 of both parents, thus $S = R_1 \cap R_2$. S fully encloses the blue region: $S \supset B \supset R$.

As can be seen in Figure 6.5, it cannot be guaranteed that all collisions are found. A new child does not register in the green region which is outside of the union $U = N_1 \cup N_2$. But since this volume is very small and is getting smaller or even disappears the smaller h is, this method is a good approximation and almost all collisions are avoided. Nevertheless, by using a smaller collision distance β or reducing h_{max} it is possible to detect and avoid all collisions.

Collision Handling

When we detect that a child point c collides with an existing point p and therefore is rejected, we move p into the direction of c in order to achieve that p is more equally distanced to the nearby points. This is done by averaging the positions

and normals:

$$\mathbf{x}_p = \frac{\mathbf{x}_p w_p + \mathbf{x}_c w_c}{w_p + w_c} \tag{6.16}$$

$$\mathbf{n}_p = \frac{\mathbf{n}_p w_p + \mathbf{n}_c w_c}{w_p + w_c}, \tag{6.17}$$

where w is a weight which is assigned to a point at the time of creation. We chose this weight to be proportional to the distance from its parents, where a high weight is assigned to all $p \in P_0$ in order to preserve the original surface geometry. After a collision is detected, the weight of the point which is moved in the process is adjusted as well:

$$w_p = w_p + w_c. \tag{6.18}$$

This averaging positively affects the uniformity of the points after the refinement.

6.2.4 Neighborhood Update

To execute the next refinement step s_{i+1}, the new neighborhoods defined by r_{i+1} have to be determined for each existing point. Instead of using a search data structure and recomputing the neighborhoods in each step, we iteratively update the neighbors of each point which is less expensive concerning computation time and memory usage while maintaining correctness.

Neighborhood updates are done simultaneously to the registering described in Section 6.2.3 as in both steps we have to access all points $p^{s_{j<i}} \in U$. In order to save computation time, the distances which were already calculated in the collision test are reused, and neighborhoods are not updated in the last refinement step. We distinguish between three different neighbor relations (note that these are symmetric since the same r is used for all points):

1. Two points created in one of the last refinement steps $s_{j<i}$ are neighbors.

2. A point created in one of the last refinement steps $s_{j<i}$ and a point created in the current refinement step s_i are neighbors.

3. Two points created in the current refinement step s_i are neighbors.

The update of type 1 neighborhoods is straightforward: as $N_i^{s_{i+1}} \subset N_i^{s_i}$ and the distance to each neighbor is already known from the last refinement step, we can update the neighborhoods cheaply by comparing the distance to r_{i+1}.

Neighbor relations of type 2 are generated after having checked a child point for collisions. When a child does not collide with any other point and therefore is accepted, it is added to the neighborhood of all $p^{s_{j<i}} \in U | d \leq r_{i+1}$, and vice versa.

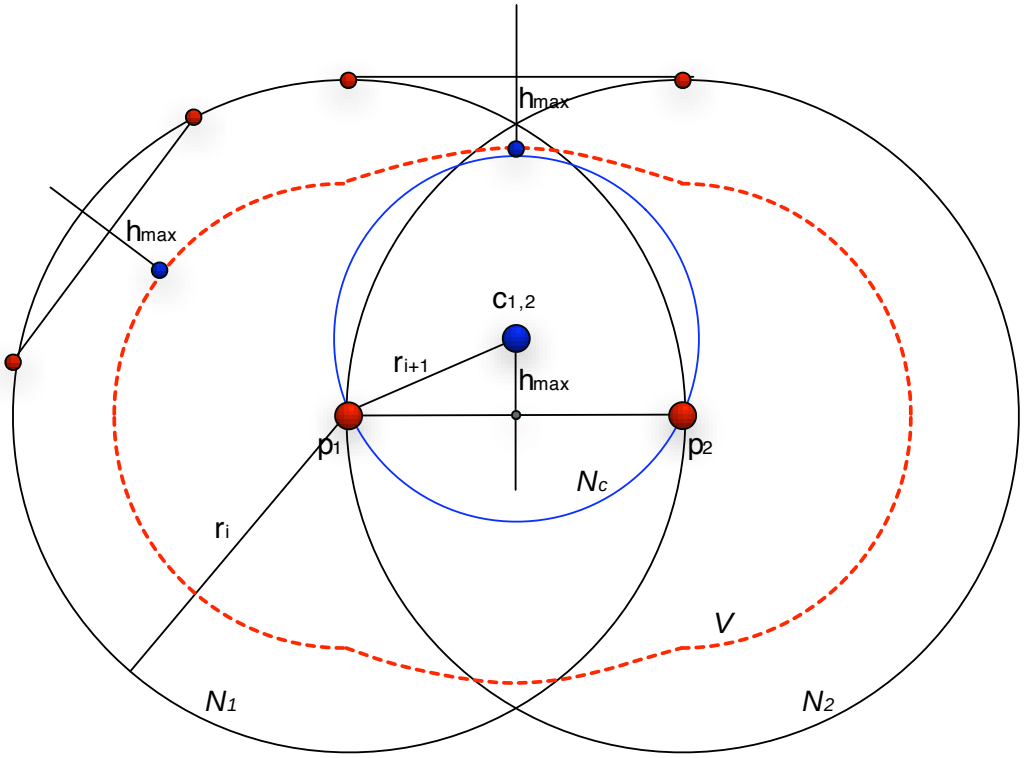


Figure 6.7: *A point created by two parent points outside U are always outside the red region V. As V fully encloses the child's neighborhood N_c of the next refinement step (blue) all candidate neighbor points are known and neighborhoods can be updated correctly.*

Additionally, the child has to be added to all points $p^{s_i} | d \leq r_{i+1}$ (type 3 relations). Figure 6.7 illustrates that it is sufficient to know all $p^{s_{j<i}} \in U$ to find all neighbor candidate points p^{s_i}, as long as each point $p^{s_{j<i}}$ knows the added children where it was involved as a parent, and therefore correct neighborhoods can be guaranteed. As we know h_{max}, r_{i+1}, and the maximal distance between two parent points $d_{max} = r_i$, we can show geometrically that it is not possible that two points $p^{s_{j<i}}$ both outside of U are refined yielding a new point which is closer than r_{i+1} to the child $c_{1,2}$. All points created by such parents would lie outside of the red dashed region V illustrated in Figure 6.7. As can be seen, V completely encloses the neighborhood N_c^{i+1} defined by r_{i+1} (illustrated in blue), even in the most extreme case where $h_{child} = h_{max}$: $V \supset N_c^{i+1}$. This means that all neighbor candidate points p^{s_i} are known by the child and can be distance checked.

6.3 Surface Reconstruction

A challenge with particle methods is to generate a smooth renderable surface from the resulting set of points. There are many different approaches for solving this problem, one was presented in [Premoze et al., 2003] where a level-set simulation guided by particles is used. Another method is presented in [Müller et al., 2003] where the surface reconstruction is based on a color function, and [Wald and Seidel, 2005] presented a method which leads to smooth surfaces, but which depends on high quality normals at every point. Each of the above approaches has advantages and disadvantages concerning quality and speed. Recently, [Zhu and Bridson, 2005] presented an approach, where the center of mass is taken into account without the need of normals. They achieve very smooth surfaces, but their technique leads to significant artifacts in concave regions and between isolated particles and splashes. It is proposed to remove these artifacts in a postprocessing step. In our work we propose a modification which uses a detector for errors located in concave regions and corrects them on the fly.

The implicit function proposed by [Zhu and Bridson, 2005] is defined as

$$\phi(\mathbf{r}) = |\mathbf{r} - \overline{\mathbf{r}}(\mathbf{r})| - R, \tag{6.19}$$

where $\overline{\mathbf{r}}(\mathbf{r})$ is the center of mass of a query point's neighborhood and R can be interpreted as a desired distance of the surface from the particles. The problem with the use of $\overline{\mathbf{r}}(\mathbf{r})$ is that it can happen that we get a center of mass which erroneously ends up outside of the surface to be reconstructed in concave regions or between near but separated particles. Examining the changes of $\overline{\mathbf{r}}(\mathbf{r})$ when moving the query point $\mathbf{r}$, one can observe that in problematic situations $\overline{\mathbf{r}}(\mathbf{r})$ changes substantially faster than the corresponding $\mathbf{r}$ (Figure 6.8). To determine how $\overline{\mathbf{r}}(\mathbf{r})$ changes we investigate $\nabla_{\mathbf{r}}(\overline{\mathbf{r}}(\mathbf{r}))$. This 3x3 matrix specifies how small changes in $\mathbf{r}$ translate into a change of $\overline{\mathbf{r}}(\mathbf{r})$. Since we are interested in detecting fast movements we check the largest Eigenvalue EV_{max}.

We define the implicit surface function as

$$\phi(\mathbf{r}) = |\mathbf{r} - \overline{\mathbf{r}}(\mathbf{r})| - Rf, \tag{6.20}$$

where f is a factor $\in [0..1]$. $\overline{\mathbf{r}}(\mathbf{r})$ is given by

$$\overline{\mathbf{r}}(\mathbf{r}) = \frac{\sum_j \mathbf{r} W(|\mathbf{r} - \mathbf{r_j}|, ir)}{\sum_j W(|\mathbf{r} - \mathbf{r_j}|, ir)}, \tag{6.21}$$

where ir is the influence radius used in the visualization, defining the smoothness of the surface, and W is the density kernel function.

We compare the largest Eigenvalue EV_{max} to two previously defined thresholds t_{low} and t_{high}. We adjust f and with this the resulting distance of the surface

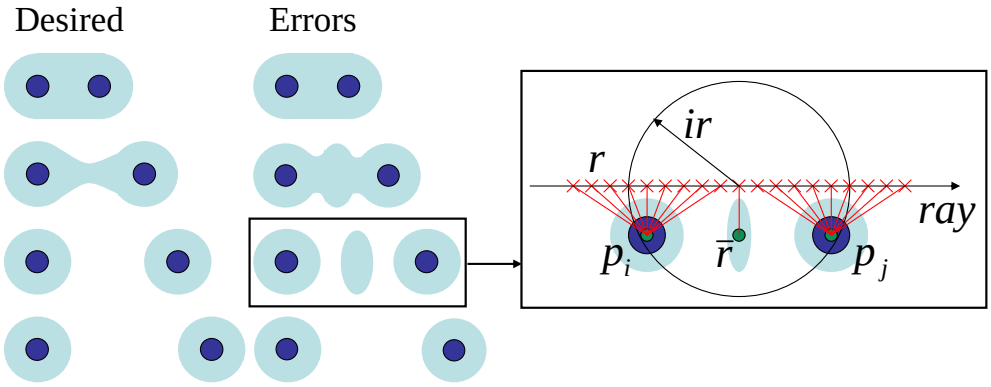


Figure 6.8: *Surface reconstruction errors in the method presented by [Zhu and Bridson, 2005] before postprocessing: when particles are separated by distances comparable to the influence radius, the center of mass may move very quickly leading to surface artifacts.*

from the particles according to the following rule, which makes sure that the first and the second order derivatives are smooth, in order to avoid hard transitions on the surface (Figure 6.9):

$$f = \begin{cases} 1 & EV_{max} < t_{low} \\ \gamma^3 - 3\gamma^2 + 3\gamma & \text{otherwise} \end{cases} \tag{6.22}$$

$$\gamma = \frac{t_{high} - EV_{max}}{t_{high} - t_{low}}. \tag{6.23}$$

Note that vanishing derivatives are not required around $EV_{max} = t_{high}$ since in these situations the surface is contracted to one point anyway as the resulting f will then be zero. With this modification the reconstructed surface avoids most of the errors without sacrificing the smoothness which is particularly important if a simulation consists only of a sparse set of particles.

We use our surface reconstruction on the fly during the rendering process. For this purpose, we have adapted the raytracer Povray (http://www.povray.org) in such a way that it can directly raytrace our particles.

6.4 Results

We have tested our new refinement method on different irregularly sampled point scenes and rendered them either using the raytracing approach presented in Section 6.3 or point splatting. All timings are given for an Intel Core2 2.66 GHz.

First, we applied our refinement method to the static ball joint point model with pre-computed, high quality normals. We have randomly chosen 9K points

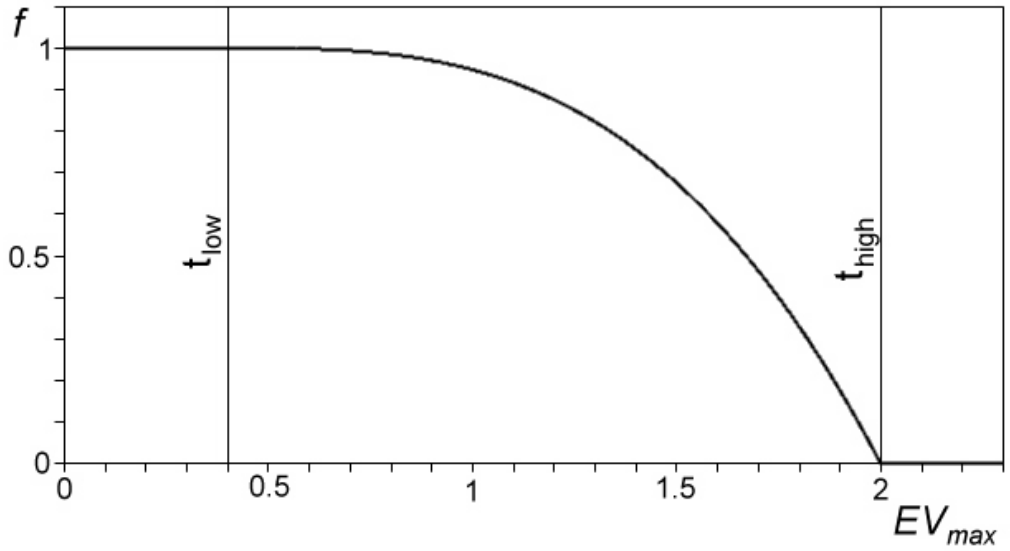


Figure 6.9: *Plot of the factor f (Equation (6.22)) using the thresholds $t_{low} = 0.4$ and $t_{high} = 2.0$. The closer EV_{max} is to t_{high}, the smaller is the resulting distance of the surface to the particles.*

of the original model consisting of 140K points. Then, we have refined the points yielding 50K points (Figure 6.10). The initial and the upsampled points are visualized using point splatting. Although surface details seem to be lost with the randomly chosen point set, they can be accurately recovered by applying our upsampling method. Furthermore, artifacts at the silhouette can be reduced. The quality of the splatting could be further improved by optimally determining the splat radius for each point, whereas in our examples we use a constant radius for all points.

Two frames of a low-resolution splashing column simulation consisting of 3K physics particles are shown in Figure 6.11. During the whole simulation, a point generation rate between 415K and 845K points per second is achieved (Table 6.1). A simulation sequence running at interactive rates is demonstrated in Figure 6.12, where the upsampled fluid with 14K surface points is running at 11fps (41 time steps per second) and the initial fluid with 1K surface points at 17fps (58 time steps per second). These timings include all computational costs (physics, normals, refinement, and visualization).

The effect of upsampling a textured fluid can be seen in Figure 6.13. Whereas the initial fluid appears blurry, the upsampled fluid is much sharper and detail-conserving. The introduced smoothing related to the original texture is visualized in the bottom row, where red and yellow correspond to low and high smoothing,

	P_0	P_3	$t_{refine}[s]$	$points/s$
Initial block	1'208	42'342	0.05	846'840
Splashing	1'916	103'840	0.25	415'360
Equilibrium	1'359	35'547	0.07	507'814

Table 6.1: *Point numbers and performances of 3 individual frames of the column splashing simulation sequence.*

respectively.

6.5 Discussion

Currently, we reach a point generation rate of up to 34K points per frame at 25fps. It is possible to improve the computation times or the visual quality even further by integrating more sophisticated methods to optimally select the points which are going to be refined. Selection operators based on curvature and level of detail information could be applied, additionally, the computation costs of the refinement could be approximately halved by omitting the upsampling of occluded points. While we present the performance of our algorithm on irregular point samples it is to be noted that it can be easily applied to regular point samples as well. In fact, regular point samples facilitate the point generation process and actually improve the performance as the refinement radii can be smaller, eliminating many of the potential point collisions.

Up to now, our refinement technique does not make use of temporal coherence. This may lead to problems in splashing areas where isolated particles merge and split and the rendered topology might undergo sudden changes. Although the integration of time-coherent aspects would reduce this problem, it would come at the expense of processing time as connectivity information would need to be stored and reused in each simulation step. As processing speed was one of our major constraint, temporal coherence is currently not integrated in our implementation.

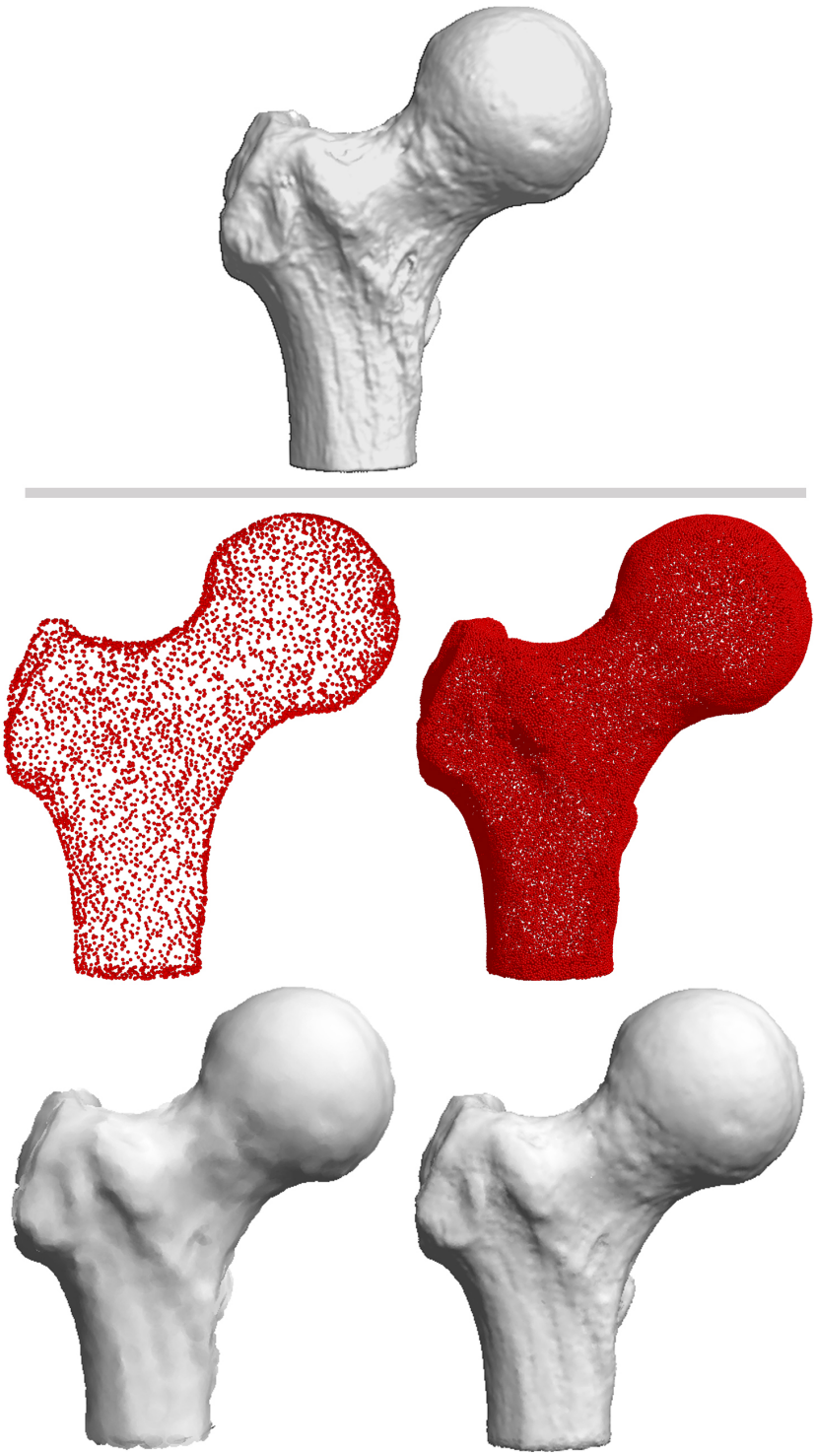


Figure 6.10: *Top row: splatted surface of the original point model (140K). Left: 9K randomly chosen points. Right: random point set upsampled to 50K points.*

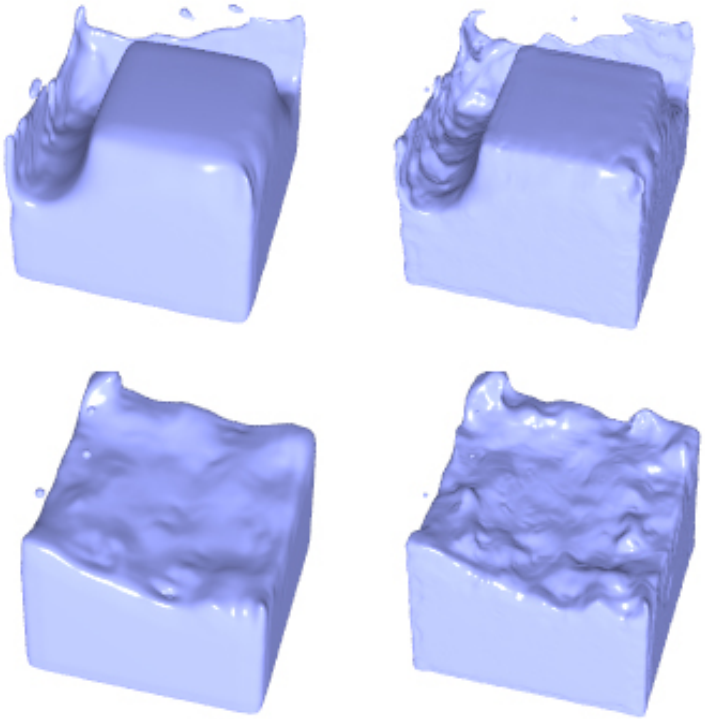


Figure 6.11: *Three frames of the column splashing simulation. The left and right image of each pair show the raytraced surface of the inital surface points and the surface after 3 refinement steps, respectively.*

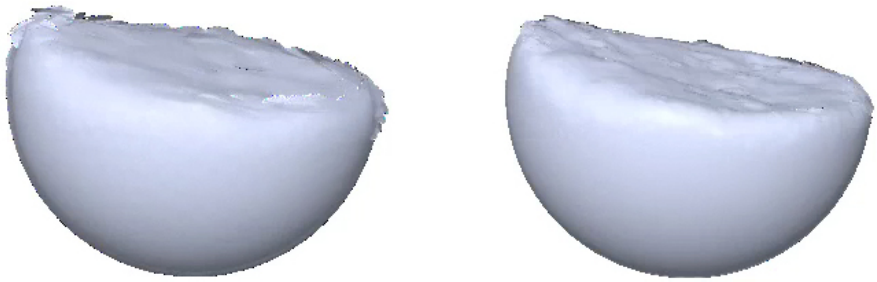


Figure 6.12: *Initial and refined points (left: 1K, right: 14K) simulated and rendered at 17fps and 11fps, respectively.*

Figure 6.13: *Left: inital points (12K). Right: upsampled points (140K). Bottom row: smoothing related to the original texture.*

CHAPTER 7

CONCLUSIONS

7.1 Summary

In this thesis, new methods are presented which can efficiently and effectively solve some of the major problems of particle-based fluid solvers, in particular of the Lagrangian SPH model. By using our methods, the physical behavior and the resulting visual quality of liquids are improved, thus rendering particle-based fluid simulations more attractive for graphics applications.

Perhaps the most critical point of particle simulation models is the difficulty to enforce incompressibility. While the grid data structure in Eulerian models facilitates the formulation and solving of the linear equation system, the unstructured particle configurations in Lagrangian models lead to more complex equation systems and higher computation costs. Although incompressibility can also be enforced by using a stiff equation of state as it is done in WCSPH, the severe time step restriction as well as the difficulties to determine the appropriate stiffness parameter represent major issues and thus compressible systems are often favored despite compression artifacts. In this thesis, we proposed a novel incompressible SPH solver which combines the advantages of both WCSPH and ISPH in one model, namely low computational cost per physics update step and large time steps. Our method includes a convergence loop which is executed in each physics step including a prediction and correction step. In each convergence iteration, the new particle positions and their densities are predicted and the variations from the reference density are computed. We derived a formulation which relates the

density fluctuation and the pressure, to reduce the density errors and to approach incompressibility. With this method, we gained a speed-up of more than one order of magnitude over the commonly used WCSPH method and we showed that the simulation results are in good agreement with WCSPH.

One strength of particles is their flexibility to simulate complex interaction effects between fluids and solids as well as between multiple different fluids. There is no need to track the surface or interface of the fluid as it has to be done in Eulerian fluid solvers. However, the standard SPH approach shows severe artifacts at interfaces when simulating multiple fluids with density ratios. These deficiencies include spurious, severe interface tension and instabilities, preventing an animator to simulate turbulently mixing fluids or to control the tension acting between the liquids. In this thesis, we presented modifications of the SPH formulation which correct for density problems, spurious and unphysical interface tension, and instabilities otherwise present at high density contrast interfaces. High density ratios can now be simulated stably, and the fluid behavior can be controlled according to the simulation problem of interest. The modification is easy to implement and does not require smaller time steps than the original SPH method.

In order to make the interaction between fluids and solids more flexible and simpler to handle, we presented a unified SPH model for the simulation of a wide variety of fluid-solid interaction processes and effects. To achieve this, we modified previous fluid and solid simulation models and integrated the different methods into a single one. The use of a unified method renders an interface between the fluid and solid models unnecessary. This simplifies the interaction between the different objects, and new effects like solidification of hot fluid matter inside cold liquid or solidification of liquid on a cold object are made possible. In addition, splitting and merging due to phase change processes can be achieved, while the distinction of touching objects where no melting is involved is presented.

Interactive applications like games or medical simulators have different requirements for a fluid simulation than offline applications like feature films or commercials. Often, the simulation is run on a single workstation at 25 fps, limiting the resolution and the complexity of the underlying physics simulation. The coarse representation comes at the cost of degraded visual quality since the number of particles has to be small. In this thesis, we have presented a refinement method which is suitable to efficiently improve the visual quality of low resolution particle-based fluids for interactive applications. Our algorithm is able to robustly detect and refine surface points, and does not require any pre-processing. Due to our fast neighborhood update we can reduce computation and memory costs. Our interpolation method can handle complex surfaces and splashes, and features like edges can be preserved. Adding points too close to existing points is avoided yielding a nearly uniform point distribution. Our algorithm can generate up to 34k points per frame at 25fps which makes it suitable for interactive fluid

applications.

Additionally, we have presented a surface reconstruction technique that avoids a surface mesh extraction but that directly raytraces the particles. One difficulty is to achieve smooth surfaces from the particles by avoiding bumps due to the irregular particle distribution. In this thesis, we introduced a new surface definition based on the center of mass of the particles. Our method does not depend on high quality point normals which avoids the problems of inaccurate normals in splashes and droplets. Furthermore, we demonstrated how artifacts in concave regions can be avoided by detecting such problems on the fly and adapting the surface accordingly.

7.2 Directions For Future Work

Although we succeeded to solve some of the major problems of the SPH particle model, other issues are still present and should be tackled in the future. The challenges we encountered during this thesis are summarized in the following.

- *Boundary deficiency*

  The particle deficiency at the boundary implicates problems in the density computation and thus in the resulting pressure and force fields. With the standard density summation equation, densities of particles close to the boundary are underestimated due to empty parts in the neighborhood. These underestimations cause difficulties to enforce incompressibility since in these situations particle compression is not identified properly (discussed in Chapter 3).

  As we have previously discussed, the convergence equation would solve this issue, but it has the problem of integration errors which are summed up, causing unphysical behavior and stability problems. Another solution is the inclusion of ghost particles along the boundaries. Since they are included in the density computation they would compensate for the empty parts in the neighborhood. However, the creation of ghost particles along complex boundaries is a difficult task, and the computation costs increase heavily since they have to be included in the neighbor search and density computation. The most promising technique seems to be the corrected SPH method which uses adapted kernel functions for the density calculation. Since the particle volume is constant when incompressibility is enforced, the computational overhead is negligible [Becker et al., 2009].

- *Viscous fluids*

  Although enforcing incompressibility does improve the splashing behavior of fluids, liquids still have some undesired viscosity compared to real liquids. We believe that this undesired viscosity has the following reasons:

  First, the viscous force which has to be added to stabilize the particle system induces undesired damping. Approaches have been made to reduce the viscosity, for example by using an adaptive strength depending on the velocity field. Although these methods can reduce the viscous behavior, further studies and comparisons of the different formulations have to be made.

  Second, SPH smoothes the physical quantities over a certain area. This smoothing induces some viscosity, particularly visible in low-resolution simulations. The higher the particle number, the smaller the smoothing effect and the better the resulting physical behavior of the fluid. This can be nicely seen when comparing the left columns of Figures 3.4 and 3.5.

  Third, when simulating multiple fluids, one can notice that the behavior of small, light volumes is negatively affected as the buoyancy is damped in specific situations (see Chapter 4). This defect is even visible without integrating any viscous forces into the model. We believe that this defect results from pressure forces compelling the particles to arrange in a stable equilibrium lattice structure. As a result, the buoyant volumes have to break open the crystallized particle configuration in order to rise. The result is that the fluid appears to be too viscous.

- *Resolution independency of fluids and solids*

  While a fluid has to be represented by at least several 100k particles to achieve good physical and visual results, interaction effects with coarsely sampled solid bodies still produce plausible interaction effects. Thus, it is desirable to have a simulation model which allows different resolutions of different bodies, i.e. high resolution fluids and lower resolution solids. Although our unified model described in Chapter 5 has shown to be very flexible and able to stably simulate melting and solidification effects, it depends on equivalent particle resolutions of fluids and solids. In order to improve the efficiency of the simulator, the presented model has to be modified to overcome this resolution dependency.

- *Efficiency*

  As indicated above, a certain particle resolution is necessary to achieve good physical and visual results. The higher the particle number, the more details

like splashes, droplets, and breaking waves can be simulated. Recent research papers have presented single CPU simulations of approx. 1-3M particles, but we desire to increase the resolution by at least a factor of 10 in the future to keep up with recent Eulerian simulations. In order to cope with the computational burden new algorithms and implementations have to be explored. One possibility to speed up the simulation is to make use of parallel machines and hardware implementations. Alternatively, new fluid representations can be explored, like adaptive particle resolutions where more particles are used in turbulent areas, or camera dependent level of detail. Furthermore, the dimension can be reduced from 3D to 2D at inactive regions, for example by coupling a 3D SPH simulation with 2D heightfield fluids.

- *Comparison with Eulerian solvers*

  Since there is still a large gap between the currently used particle resolutions and grid resolutions it is hard to compare the visual results of these solvers with each other. Furthermore, it has to be identified how the number of cells and the number of particles correspond. With this knowledge, the performance as well as the visual results can be better compared with each other, allowing to identify the strengths and weaknesses of both methods more clearly. This information will help an animator or modeler to evaluate the adequate method to solve a particular problem.

During this dissertation, the particle model SPH has proven to be a powerful fluid solver which facilitates the simulation of various complex interaction effects as well as detailed structures like splashes and sprays. With the new methods we have presented in this thesis, some of the major drawbacks of SPH and particles in general can be solved. However, certain limitations of SPH still remain, but we hope that our work inspires researchers to start or continue working on particle-based fluids in the future. To conclude, in this thesis we often experienced the advantageous properties of the particle representation, but we believe that Lagrangian and Eulerian fluid solvers have to be compared more thoroughly in the future to evaluate the advantages and deficiencies of each particular solver more clearly.

BIBLIOGRAPHY

[Adams and Dutre, 2003] Adams, B. and Dutre, P. (2003). Interactive boolean operations on surfel-bounded solids. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 22(3):651–656.

[Adams et al., 2007] Adams, B., Pauly, M., Keiser, R., and Guibas, L. J. (2007). Adaptively sampled particle fluids. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 26(3):48–54.

[Adamson and Alexa, 2003] Adamson, A. and Alexa, M. (2003). Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics Symposium on Geometry Processing*, pages 230–239.

[Ageia, 2005] Ageia (2005). Physics, gameplay and the physics processing unit. White paper.

[Agertz et al., 2006] Agertz, O., Moore, B., Stadel, J., Potter, D., Miniati, F., Read, J., Mayer, L., Gawryszczak, A., Kravtsov, A., Monaghan, J., Nordlund, A., Pearce, F., Quilis, V., Rudd, D., Springel, V., Stone, J., Tasker, E., Teyssier, R., Wadsley, J., and Walder, R. (2006). Fundamental differences between SPH and grid methods. *Mon. Not. R. astr. Soc.*, astro-ph/0610051.

[Alexa et al., 2003] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. (2003). Computing and rendering point set surfaces. *IEEE Transactions on Computer Graphics and Visualization*, 1(9):3–15.

[Balsara, 1995] Balsara, D. S. (1995). Von neumann stability analysis of Smoothed Particle Hydrodynamics—suggestions for optimal algorithms. *J. Comput. Phys.*, 121(2):357–372.

[Baraff, 1997] Baraff, D. (1997). An introduction to physically based modeling: Rigid body simulation 1 - unconstrained rigid body dynamics. SIGGRAPH Course Notes.

[Batchelor, 1967] Batchelor, G. K. (1967). *An Introduction to Fluid Dynamics*. Cambridge University Press.

[Becker and Teschner, 2007] Becker, M. and Teschner, M. (2007). Weakly compressible SPH for free surface flows. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 209–217.

[Becker et al., 2009] Becker, M., Tessendorf, H., and Teschner, M. (2009). Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):493–503.

[Blinn, 1982] Blinn, J. (1982). A generalization of algebraic surface drawing. In *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, pages 235–256.

[Bonet and Kulasegaram, 2002] Bonet, J. and Kulasegaram, S. (2002). A simplified approach to enhance the performance of Smoothed Particle Hydrodynamics methods. *Appl. Math. Comput.*, 126(2-3):133–155.

[Cani and Desbrun, 1997] Cani, M. P. and Desbrun, M. (1997). Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50.

[Carlson et al., 2002] Carlson, M., Mucha, P. J., Horn, R. B. V., and Turk, G. (2002). Melting and flowing. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 167–174.

[Carlson et al., 2004] Carlson, M., Mucha, P. J., and Turk, G. (2004). Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 23(3):377–384.

[Carr et al., 2001] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R. (2001). Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of ACM SIGGRAPH*, pages 67–76.

[Chen et al., 1999] Chen, J. K., Beraun, J. E., and Jih, C. J. (1999). An improvement for tensile instability in smoothed particle hydrodynamics. *J. Comput. Mech.*, 23(4):279–287.

[Chentanez et al., 2006] Chentanez, N., Goktekin, T. G., Feldman, B. E., and O'Brien, J. F. (2006). Simultaneous coupling of fluids and deformable bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 83–89.

[Colagrossi and Landrini, 2002] Colagrossi, A. and Landrini, M. (2002). Numerical simulation of interfacial flows by Smoothed Particle Hydrodynamics. *J. Comput. Phys.*, 191(2):448–475.

[Courant et al., 1967] Courant, R., Friedrichs, K., and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM J.*, 11:215–234.

[Cummins and Rudman, 1999] Cummins, S. J. and Rudman, M. (1999). An SPH projection method. *J. Comput. Phys.*, 152(2):584–607.

[Desbrun and Cani, 1996] Desbrun, M. and Cani, M. P. (1996). Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation*, pages 61–76.

[Enright et al., 2002] Enright, D., Marschner, S., and Fedkiw, R. (2002). Animation and rendering of complex water surfaces. In *Proceedings of Computer graphics and interactive techniques*, pages 736–744.

[Feldman et al., 2005] Feldman, B. E., OBrien, J. F., and Klingner, B. M. (2005). Animating gases with hybrid meshes. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 24(3):904–909.

[Foster and Fedkiw, 2001] Foster, N. and Fedkiw, R. (2001). Practical animation of liquids. In *Proceedings of Computer graphics and interactive techniques*, pages 23–30. ACM Press.

[Génevaux et al., 2003] Génevaux, O., Habibi, A., and Dischler, J. M. (2003). Simulating fluid-solid interaction. In *Graphics Interface*, pages 31–38.

[Gingold and Monaghan, 1977] Gingold, R. A. and Monaghan, J. J. (1977). Smoothed Particle Hydrodynamics: theory and application to non-SPHerical stars. *Mon. Not. R. astr. Soc.*, 181:375–389.

[Goktekin et al., 2004] Goktekin, T. G., Bargteil, A. W., and O'Brien, J. F. (2004). A method for animating viscoelastic fluids. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 23(3):463–468.

[Greenwood and House, 2004] Greenwood, S. T. and House, D. H. (2004). Better with bubbles: enhancing the visual realism of simulated fluid. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 287–296.

[Guendelman et al., 2005] Guendelman, E., Selle, A., Losasso, F., and Fedkiw, R. (2005). Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 24(3):973–981.

[Guennebaud et al., 2004] Guennebaud, G., Barthe, L., and Paulin, M. (2004). Dynamic surfel set refinement for high quality rendering. *Computer and Graphics*, 28(6):827–838.

[Guennebaud et al., 2005] Guennebaud, G., Barthe, L., and Paulin, M. (2005). Interpolatory refinement for real-time processing of point-based geometry. *Computer Graphics Forum (Proceedings of Eurographics)*, 24(3):657–667.

[Guennebaud and Gross, 2007] Guennebaud, G. and Gross, M. (2007). Algebraic point set surfaces. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 26(3):23.

[Harada et al., 2007] Harada, T., Koshizuka, S., and Kawaguchi, Y. (2007). Smoothed Particle Hydrodynamics on GPUs. In *Proceedings of Computer Graphics International*, pages 63–70.

[Hernquist and Katz, 1989] Hernquist, L. and Katz, N. (1989). TreeSPH: A Unification of SPH with the Hierarchical Tree Method. *ApJS*, 70:419–446.

[Hong and Kim, 2003] Hong, J. M. and Kim, C. H. (2003). Animation of bubbles in liquid. *Computer Graphics Forum*, 22(3):253–262.

[Hong and Kim, 2005] Hong, J. M. and Kim, C. H. (2005). Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH Proceedings)*, 24(3):915–920.

[Hoover, 1998] Hoover, W. G. (1998). Isomorphism linking smooth particles and embedded atoms. *Physics A*, 260:244–254.

[Hu and Adams, 2006] Hu, X. Y. and Adams, N. A. (2006). A multi-phase SPH method for macroscopic and mesoscopic flows. *J. Comput. Phys.*, 213(2):844–861.

[Hu and Adams, 2007] Hu, X. Y. and Adams, N. A. (2007). An incompressible multi-phase SPH method. *J. Comput. Phys.*, 227(1):264–278.

[Kang et al., 2000] Kang, M., Fedkiw, R. P., and Liu, X.-D. (2000). A boundary condition capturing method for multiphase incompressible flow. *J. Sci. Comput.*, 15(3):323–360.

[Keiser, 2006] Keiser, R. (2006). *Meshless Lagrangian Methods for Physics-Based Animations of Solids and Fluids*. PhD thesis, ETH Zurich.

[Keiser et al., 2005] Keiser, R., Adams, B., Gasser, D., Bazzi, P., Dutre, P., and Gross, M. (2005). A unified Lagrangian approach to solid-fluid animation. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 125–133.

[Lenaerts et al., 2008] Lenaerts, T., Adams, B., and Dutré, P. (2008). Porous flow in particle-based fluid simulations. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 27(3):1–8.

[Levin, 2003] Levin, D. (2003). Mesh-independent surface interpolation. *In Geometric Modeling for Scientific Visualization*, pages 37–49.

[Liu et al., 2005] Liu, J., Koshizuka, S., and Oka, Y. (2005). A hybrid particle-mesh method for viscous, incompressible, multiphase flows. *J. Comput. Phys.*, 202(1):65–93.

[Lombardi et al., 1999] Lombardi, J. C., Sills, A., Rasio, F. A., and Shapiro, S. L. (1999). Tests of spurious transport in Smoothed Particle Hydrodynamics. *J. Comput. Phys.*, 152(2):687–735.

[Losasso et al., 2006a] Losasso, F., Irving, G., Guendelman, E., and Fedkiw, R. (2006a). Melting and burning solids into liquids and gases. *IEEE TVCG*, 12:343–352.

[Losasso et al., 2006b] Losasso, F., Shinar, T., Selle, A., and Fedkiw, R. (2006b). Multiple interacting liquids. *ACM Trans. Graph. (SIGGRAPH Proceedings)*, 25(3):812–819.

[Losasso et al., 2008] Losasso, F., Talton, J., Kwatra, J., and Fedkiw, R. (2008). Two-way coupled SPH and particle level set fluid simulation. *IEEE TVCG*, 14(4):797–804.

[Lucy, 1977] Lucy, L. B. (1977). A numerical approach to testing of the fission hypothesis. *Astron. J.*, 82:1013–1024.

[Mao and Yang, 2006] Mao, H. and Yang, Y.-H. (2006). Particle-based immiscible fluid-fluid collision. In *Proceedings of Graphics Interface 2006*, pages 49–55.

[Mihalef et al., 2006] Mihalef, V., Unlusu, B., Metaxas, D., Sussman, M., and Hussaini, M. Y. (2006). Physics based boiling simulation. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 317–324.

[Molemaker et al., 2008] Molemaker, J., Cohen, J., Patel, S., and Noh, J. (2008). Low viscosity flow simulations for animation. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 9–18.

[Monaghan, 1989] Monaghan, J. J. (1989). On the problem of penetration in particle methods. *J. Comput. Phys.*, 81:1–15.

[Monaghan, 1992] Monaghan, J. J. (1992). Smoothed Particle Hydrodynamics. *Annu. Rev. Astron. Physics*, 30:543.

[Monaghan, 1994] Monaghan, J. J. (1994). Simulating free surface flows with SPH. *J. Comput. Phys.*, 110:399–406.

[Monaghan, 2005] Monaghan, J. J. (2005). Smoothed Particle Hydrodynamics. *Rep. Prog. Phys.*, 68:1703–1759.

[Morris, 2000] Morris, J. P. (2000). Simulating surface tension with Smoothed Particle Hydrodynamics. *International Journal for Numerical Methods in Fluids*, 33:333–353.

[Morris et al., 1997] Morris, J. P., Fox, P. J., and Zhu, Y. (1997). Modeling low Reynolds number incompressible flows using SPH. *J. Comput. Phys.*, 136:214.

[Morris and Monaghan, 1997] Morris, J. P. and Monaghan, J. J. (1997). A switch to reduce SPH viscosity. *J. Comput. Phys.*, 136(1):41–50.

[Mullen et al., 2009] Mullen, P., Crane, K., Pavlov, D., Tong, Y., and Desbrun, M. (2009). Energy-preserving integrators for fluid animation. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 28(3):to appear.

[Müller et al., 2003] Müller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 154–159.

[Müller et al., 2005a] Müller, M., Heidelberger, B., Teschner, M., and Gross, M. (2005a). Meshless deformations based on shape matching. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 24(3):471–478.

[Müller et al., 2004] Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., and Alexa, M. (2004). Point based animation of elastic, plastic and melting objects. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 141–151.

[Müller et al., 2007] Müller, M., Schirm, S., and Duthaler, S. (2007). Screen space meshes. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 9–15.

[Müller et al., 2004] Müller, M., Schirm, S., Teschner, M., Heidelberger, B., and Gross, M. (2004). Interaction of fluids with deformable solids. *Journal of Computer Animation and Virtual Worlds*, 15(3-4):159–171.

[Müller et al., 2005b] Müller, M., Solenthaler, B., Keiser, R., and Gross, M. (2005b). Particle-based fluid-fluid interaction. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 237–244.

[Nealen et al., 2005] Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. (2005). Physically based deformable models in computer graphics. In *Eurographics State of the Art Report*, pages 71–94.

[O'Brien et al., 2002] O'Brien, J. F., Bargteil, A. W., and Hodgins, J. K. (2002). Graphical modeling and animation of ductile fracture. In *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, volume 21, pages 291–294.

[Ohtake et al., 2003] Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., and Seidel, H. P. (2003). Multi-level partition of unity implicits. *ACM Transaction on Graphics*, 3(22):463–470.

[Ott and Schnetter, 2003] Ott, F. and Schnetter, E. (2003). A modified SPH approach for fluids with large density differences. arXiv:physics/0303112.

[Pajarola, 2003] Pajarola, R. (2003). Efficient level-of-details for point based rendering. In *Proceedings IASTED International Conference on Computer Graphics and Imaging*.

[Pauly et al., 2002] Pauly, M., Gross, M., and Kobbelt, L. P. (2002). Efficient simplification of point-sampled surfaces. In *Proceedings of IEEE Visualization*, pages 163–170.

[Pauly et al., 2005] Pauly, M., Keiser, R., Adams, B., Dutre, P., Gross, M., and Guibas, L. J. (2005). Meshless animation of fracturing solids. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 24(3):957–964.

[Pauly et al., 2003] Pauly, M., Keiser, R., Kobbelt, L. P., and Gross, M. (2003). Shape modeling with point-sampled geometry. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 22(3):641–650.

[Pauly et al., 2006] Pauly, M., Kobbelt, L. P., and Gross, M. (2006). Point-based multiscale surface representation. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 25(2):177–193.

[Pelupessy et al., 2003] Pelupessy, F. I., Schaap, W. E., and van de Weygaert, R. (2003). Density estimators in particle hydrodynamics: DTFE versus regular SPH. *Astronomy and Astrophysics*, 403:389–398.

[Premoze et al., 2003] Premoze, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R. T. (2003). Particle-based simulation of fluids. In *Proceedings of Eurographics*, pages 401–410.

[Selle et al., 2005] Selle, A., Rasmussen, N., and Fedkiw, R. (2005). A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 24(3):910–914.

[Shao, 2006] Shao, S. (2006). Incompressible SPH simulation of wave breaking and overtopping with turbulence modelling. *Int. J. Numer. Meth. Fluids*, 50:597–621.

[Solenthaler and Pajarola, 2008] Solenthaler, B. and Pajarola, R. (2008). Density contrast SPH interfaces. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 211–218.

[Solenthaler and Pajarola, 2009] Solenthaler, B. and Pajarola, R. (2009). Predictive-corrective incompressible SPH. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 28(3):to appear.

[Solenthaler et al., 2007a] Solenthaler, B., Schläfli, J., and Pajarola, R. (2007a). A unified particle model for fluid-solid interactions. *Journal of Computer Animation and Virtual Worlds*, 18(1):69–82.

[Solenthaler et al., 2007b] Solenthaler, B., Zhang, Y., and Pajarola, R. (2007b). Efficient refinement of dynamic point data. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 65–72.

[Stam, 1999] Stam, J. (1999). Stable fluids. In *Proceedings of Computer graphics and interactive techniques*, pages 121–128.

[Stam and Fiume, 1995] Stam, J. and Fiume, E. (1995). Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of SIGGRAPH*, pages 129–136.

[Stora et al., 1999] Stora, D., Agliati, P., Cani, M. P., Neyret, F., and Gascuel, J. (1999). Animating lava flows. In *Graphics Interface*, pages 203–210.

[Tartakovsky and Meakin, 2005] Tartakovsky, A. M. and Meakin, P. (2005). A Smoothed Particle Hydrodynamics model for miscible flow in three-dimensional fractures and the two-dimensional Rayleigh-Taylor instability. *J. Comput. Phys.*, 207(2):610–624.

[Terzopoulos et al., 1989] Terzopoulos, D., Platt, J., and Fleischer, K. (1989). Heating and melting deformable models (from goop to glop). In *Graphics Interface*, pages 219–226.

[Thürey et al., 2006] Thürey, N., Keiser, R., Pauly, M., and Rüde, U. (2006). Detail-preserving fluid control. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 7–15.

[Thürey et al., 2007] Thürey, N., Sadlo, F., Schirm, S., Müller-Fischer, M., and Gross, M. (2007). Real-time simulations of bubbles and foam within a shallow water framework. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 191–198.

[Tobor et al., 2004] Tobor, I., Reuter, P., and Schlick, C. (2004). Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. In *Proceedings of Shape Modelling International*, pages 19–30.

[Tonnesen, 1991] Tonnesen, D. (1991). Modeling liquids and solids using thermal particles. In *Graphics Interface*, pages 255–262.

[Wald and Seidel, 2005] Wald, I. and Seidel, H.-P. (2005). Interactive Ray Tracing of Point Based Models. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 9–16.

[Wicke et al., 2006] Wicke, M., Hatt, P., Pauly, M., Müller, M., and Gross, M. (2006). Versatile virtual materials using implicit connectivity. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 137–144.

[Zhang et al., 2008] Zhang, Y., Solenthaler, B., and Pajarola, R. (2008). Adaptive sampling and rendering of fluids on the GPU. In *Proceedings of the Eurographics Symposium on Volume and Point-Based Graphics*, pages 137–146.

[Zhao et al., 2006] Zhao, Y., Wang, L., Qiu, F., Kaufman, A., and Mueller, K. (2006). Melting and flowing in multiphase environment. *Computers & Graphics*, 30(4):519–528.

[Zheng et al., 2006] Zheng, W., Yong, J.-H., and Paul, J.-C. (2006). Simulation of bubbles. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 325–333.

[Zhu and Bridson, 2005] Zhu, Y. and Bridson, R. (2005). Animating sand as a fluid. *ACM Trans. Graph. (Proceedings of SIGGRAPH)*, 24(3):965–972.

[Zwicker et al., 2002] Zwicker, M., Pauly, M., Knoll, O., and Gross, M. (2002). Pointshop 3D: an interactive system for point-based surface editing. In *Proceedings of Computer graphics and interactive techniques*, pages 322–329.

CURRICULUM VITAE

Personal Information

Name	Barbara Solenthaler
Date of birth	November 10, 1978
Place of birth	Herisau, Switzerland

Education

2005 - 2009	Doctor in Informatics in the Visualization and Multimedia Lab, Department of Informatics, University of Zurich, Switzerland Subject of dissertation: ”Incompressible Fluid Simulation and Advanced Surface Handling with SPH” Advisor: Prof. Dr. Renato Pajarola, University of Zurich External Examiner: Prof. Dr. Markus Gross, ETH Zurich
2004	Dipl. Informatik-Ing. ETH, Department of Computer Science, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland Subject of diploma thesis: ”Particle-Based Fluid-Fluid Interaction” Advisor: Prof. Dr. Markus Gross, ETH Zurich

1998 - 2004	Student of Computer Science at the Swiss Federal Institute of Technology (ETH) Zurich, Switzerland
1993 - 1997	Kantonsschule Trogen, Switzerland

Publications

Journal Articles

B. Solenthaler and R. Pajarola. Predictive-Corrective Incompressible SPH. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 28(3), 2009.

B. Solenthaler, J. Schläfli, and R. Pajarola. A Unifed Particle Model for Fuid-Solid Interactions. *Computer Animation and Virtual Worlds*, 18(1):69-82, 2007.

Conference Publications

B. Solenthaler and R. Pajarola. Density Contrast SPH Interfaces. *Proceedings of ACM SIGGRAPH / EG Symposium on Computer Animation*, pp. 211-218, 2008.

Y. Zhang, B. Solenthaler, and R. Pajarola. Adaptive Sampling and Rendering of Fluids on the GPU. *Proceedings of IEEE/EG Symposium on Volume and Point-Based Graphics*, pp. 137-146, 2008.

B. Solenthaler, Y. Zhang, and R. Pajarola. Efficient Refinement of Dynamic Point Data. *Proceedings of Eurographics Symposium on Point-Based Graphics*, pp. 65-72, 2007.

M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-Based Fluid-Fluid Interaction. *Proceedings of ACM SIGGRAPH / EG Symposium on Computer Animation*, pp. 237-244, 2005.

G. Bianchi, B. Solenthaler, M. Harders, and G. Székely. Simultaneous Topology and Stiffness Identication for Mass-Spring Models based on FEM Reference Deformations. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 293-301, 2004.

Misc

Y. Zhang, B. Solenthaler, and R. Pajarola. GPU Accelerated SPH Particle Simulation and Rendering. ACM SIGGRAPH Posters, 2007.

B. Solenthaler, Y. Zhang, and R. Pajarola. Efficient Refinement of Dynamic Point Data. Posters Symposium on Computer Animation, 2007.

B. Solenthaler, J. Schläfli, and R. Pajarola. From Fluid to Solid and Back in a Unified Particle Model. Posters Symposium on Computer Animation, 2006.

B. Solenthaler. Particle-Based Fluid-Fluid Interaction. Diploma thesis, Computer Graphics Laboratory, ETH Zurich, 2004.

B. Solenthaler. Simultaneous Topology and Stiffness Identification for Mass-Spring Models based on FEM Reference Deformations. Semester thesis, Image Science Laboratory, ETH Zurich, 2004.

B. Solenthaler. Simulation of City Growth. Semester thesis, Simulation and Modeling Group, ETH Zurich, 2003.

Awards

Fritz-Kutter Award 2009 for the dissertation titled *Incompressible Fluid Simulation and Advanced Surface Handling with SPH*.